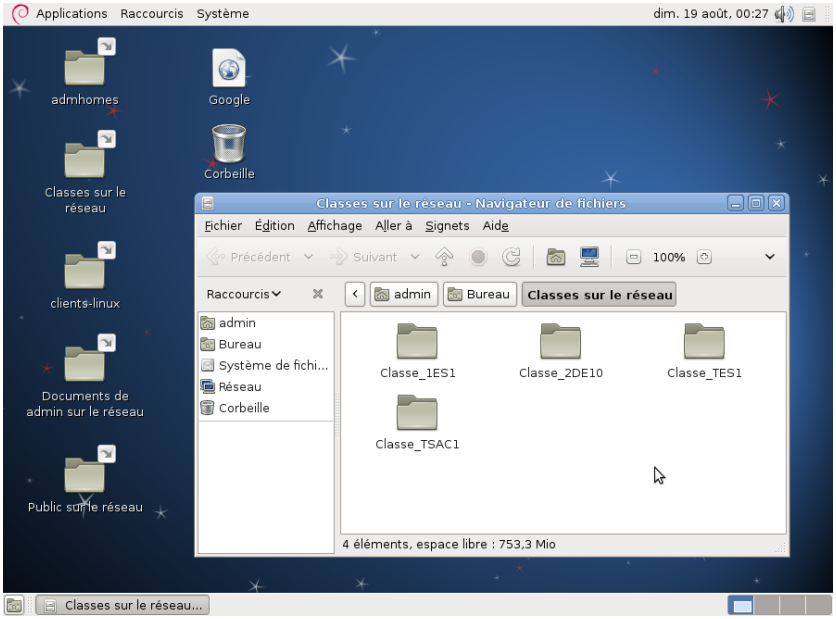
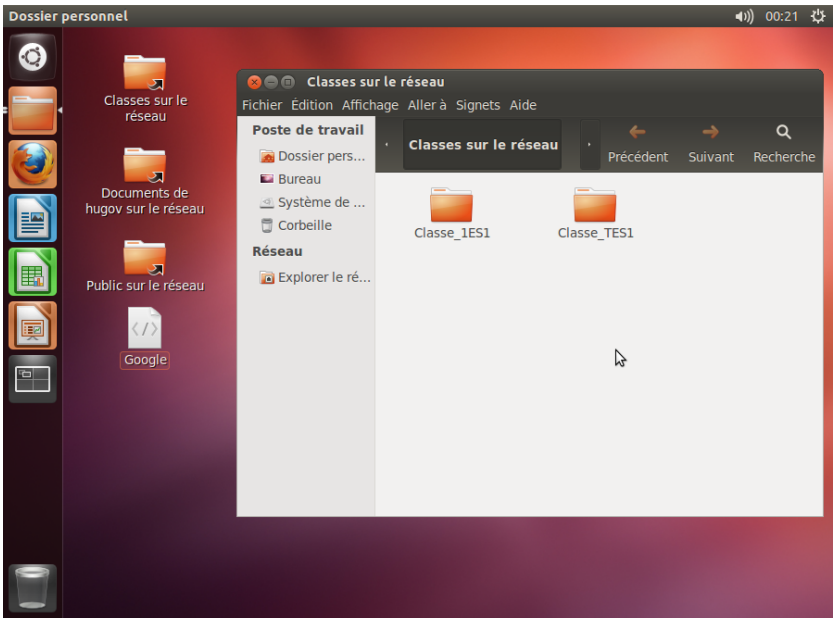
Intégration de stations de travail Debian ou Ubuntu dans un domaine SambaÉdu3 avec le paquet se3-clients-linux (version 1.1.6)



Bureau du compte **admin** du domaine sur une machine cliente Debian Squeeze.



Bureau d'un compte professeur sur une machine cliente Ubuntu Precise Pangolin.

Résumé et avertissement

Le but de ce document est de décrire un mode opératoire d'intégration de stations clientes Debian ou Ubuntu dans un domaine SambaÉdu3 (avec un serveur en version Lenny ou Squeeze) par l'intermédiaire du paquet **se3-clients-linux**. Les distributions GNU/Linux qui ont été testées sont :

- Debian Squeeze (version 6)
- Debian Wheezy (version 7)
- Ubuntu Precise Pangolin (version 12.04)
- Xubuntu Precise Pangolin (version 12.04)

Sur le long terme, l'idée serait de se borner au moins [a] à la prise en charge de toutes les versions de Debian à partir de Squeeze et de toutes les versions LTS d'Ubuntu à partir de Precise Pangolin.

En pratique, l'objectif est de pouvoir ouvrir une session sur un client Linux avec un compte du domaine et d'avoir accès à l'essentiel des partages offerts par le serveur SambaÉdu3 en fonction du compte.

Le fonctionnement de l'ensemble du paquet a été écrit de manière à tenter de minimiser le trafic réseau entre un client Linux et le serveur, notamment au moment de l'ouverture de session où la gestion des profils est très différente de celle mise en place pour les clients Windows (voir la documentation pour plus de précisions).

Avertissement : l'intégration est censée fonctionner avec les distributions ci-desssus **dans leur configuration proposée par défaut**, notamment au niveau du « display manager » [b] et de l'environnement de bureau [c]. Tout au long de la documentation, il est supposé que c'est bien le cas.

Si jamais vous tenez à changer de « display manager » sur votre distribution, il est quasiment certain que vous devrez modifier le script d'intégration de la distribution parce que celui-ci ne fonctionnera pas en l'état. Si vous tenez à changer uniquement l'environnement de bureau, il est possible que le script d'intégration fonctionne en l'état malgré tout mais nous ne pouvons en rien vous garantir le résultat final. L'apparition de régressions ici ou là par rapport à ce qui est annoncé dans ce document n'est pas à exclure.

Le 13 décembre 2013
François Lafont
Louis-Maurice De Sousa
Michel Suquet

a. Et ce serait déjà pas si mal...

b. Le « display manager » est le programme qui se lance au démarrage et qui affiche une fenêtre de connexion permettant d'ouvrir une session après authentification via un identifiant et un mot de passe. Sous Squeeze par exemple, le programme remplissant cette fonction s'appelle Gdm3 et sous Precise il s'agit de Lightdm etc...

c. L'environnement de bureau est le programme qui est lancé par le « display manager » une fois que vous vous êtes bien authentifié(e). C'est ce programme qui vous fournit le bureau que vous avez sous les yeux, un gestionnaire de fenêtres, une barre de menus etc... Les plus connus sont Gnome, KDE, Xfce, Unity, LXDE...

Table des matières

Liste des tableaux

1 Avertissement : lisez-vous la bonne documentation ?

Il est important de vous assurer que vous lisez bien la bonne documentation correspondant à la version du paquet **se3-clients-linux** qui est ou qui va être utilisée sur votre serveur. En effet, d'une version à une autre, les choses évoluent, des fonctionnalités nouvelles peuvent apparaître etc.

1.1 Si le paquet est déjà installé sur votre serveur

Dans ce cas, sur votre serveur en tant que **root**, il vous suffit de lancer la commande suivante :

```
cat /home/netlogon/clients-linux/doc/LISEZMOI.TXT
```

Cela affichera le contenu du fichier **LISEZMOI.TXT** dans lequel vous trouverez l'adresse URL vers la documentation associée à la version du paquet qui est déjà installée sur votre serveur.

1.2 Si le paquet n'est pas encore installé sur votre serveur

Et bien installez-le comme cela est expliqué à la section 2.1 page 6 puis procédez comme ci-dessus.

2 Pour les impatients qui veulent tester rapidement

2.1 Installation du paquet se3-clients-linux sur le serveur

Il faut que votre réseau local dispose d'une connexion Internet. Pour commencer, il faut préparer votre serveur Samba en y installant le paquet **se3-clients-linux**. Pour ce faire :

- Si votre serveur est sous Lenny, il faut ouvrir une console en tant que **root** et lancer :

```
apt-get update
apt-get install se3-clients-linux
```

- Si votre serveur est sous Squeeze, vous pouvez :
  - ou bien faire l'installation comme sur un serveur Lenny (en mode console donc) ;
  - ou bien faire l'installation en passant par l'interface d'administration Web du serveur via les menus **Configuration générale** → **Modules**. Dans le tableau des modules, le paquet **se3-clients-linux** correspond à la ligne avec l'intitulé **Support des clients linux**.

Attention, dans les versions précédentes du paquet, il fallait éditer le fichier **/etc/apt/sources.list** [a] et ajouter une des deux lignes suivantes :

```
# Pour un serveur en version Lenny.
deb http://francois-lafont.ac-versailles.fr/debian lenny se3

# Pour un serveur en version Squeeze.
deb http://francois-lafont.ac-versailles.fr/debian squeeze se3
```

Désormais ce n'est plus nécessaire. Le paquet est maintenant inclus dans le dépôt officiel du projet SambaÉdu, il n'est donc plus indispensable d'ajouter un dépôt supplémentaire.

Mais si vous souhaitez utiliser la toute dernière version disponible du paquet, alors il faudra dans ce cas utiliser le dépôt http://francois-lafont.ac-versailles.fr comme indiqué ci-dessus.

a. ou mieux, créer un fichier **/etc/apt/sources.list.d/se3-clients-linux.list** car le fichier **/etc/apt/sources.list** peut être réédité à votre insu lors de mises à jour du serveur.

L'installation ne fait rien de bien méchant sur votre serveur. Vous pouvez parfaitement désinstaller le paquet du serveur afin que celui-ci retrouve très exactement le même état qu'avant l'installation (voir la section 13 page 42). L'installation se borne uniquement à effectuer les tâches suivantes :

- Création d'un nouveau répertoire : le répertoire **/home/netlogon/clients-linux/**.
- Création d'un partage Samba supplémentaire sur le serveur à travers le fichier de configuration **/etc/samba/smb_CIFSFS.conf** : il s'agit du partage CIFS nommé **netlogon-linux** correspondant au répertoire **/home/netlogon/clients-linux/** du serveur.
- Lecture de certains paramètres du serveur afin d'adapter certains scripts contenus dans le paquet **se3-clients-linux** à l'environnement de votre domaine local. En fait, ces fameux paramètres récupérés lors de l'installation du paquet sont au nombre de trois :
  1. l'adresse IP du serveur ;
  2. le suffixe de base de l'annuaire LDAP ;

3. l'adresse du serveur de temps NTP.

Lors de l'installation du paquet, si jamais vous obtenez un message vous indiquant que le serveur NTP ne semble pas fonctionner, avant de passer à la suite, vous devez vous rendre sur la console d'administration Web de votre serveur (dans **Configuration générale** → **Paramètres serveur**) afin de spécifier l'adresse d'un serveur de temps qui fonctionne correctement (chose que l'on peut vérifier ensuite dans la page de diagnostic du serveur). Une fois le paramétrage effectué il vous suffit de reconfigurer le paquet en lançant, en tant que **root** sur une console du serveur, la commande suivante :

```
dpkg-reconfigure se3-clients-linux
```

Si tout se passe bien, vous ne devriez plus obtenir d'avertissement à propos du serveur NTP.

Votre serveur Samba possède donc un nouveau partage CIFS qui, au passage, ne sera pas visible par les machines clientes sous Windows. Attention, le nom du partage CIFS n'est pas le même que le nom du répertoire correspondant dans l'arborescence locale du serveur :

Nom du partage	Chemin réseau	Chemin dans l'arborescence locale du serveur
netlogon-linux	**//SERVEUR/netlogon-linux**	**/home/netlogon/clients-linux/**

Au niveau de l'installation du paquet proprement dite, côté serveur, plus aucune manipulation supplémentaire n'est nécessaire désormais.

Sachez enfin que si, pour une raison ou pour une autre, il vous est nécessaire de reconfigurer le paquet pour restaurer des droits corrects sur les fichiers, ou bien pour réadapter les scripts à l'environnement de votre serveur (parce que par exemple son IP a changé), il vous suffit de lancer la commande suivante en tant que **root** sur une console du serveur :

```
dpkg-reconfigure se3-clients-linux
```

2.2 Intégration d'un client GNU/Linux

Le répertoire **/home/netlogon/clients-linux/** de votre serveur contient un script d'intégration par type de distribution GNU/Linux. Par exemple, le script d'intégration pour des Debian Squeeze se trouve dans le répertoire :

/home/netlogon/clients-linux/distribs/squeeze/integration/

et il s'appelle **integration_squeeze.bash**. Il faudra exécuter l'un de ces scripts, en tant que **root**, **en local** sur le client GNU/Linux que vous souhaitez intégrer.

Remarque : pour copier en local sur un client GNU/Linux le script d'intégration qui se trouve sur le serveur, on pourra utiliser la bonne vieille clé USB des familles, mais on pourra aussi user et abuser de la commande **scp** (très pratique) qui permet d'effectuer très simplement des copies entre deux machines (sous GNU/Linux) distantes. Par exemple, sur le terminal d'un client Debian Squeeze, vous pourriez exécuter les commandes suivantes :

```
# Chemin du fichier sur le serveur. Le joker * nous permet simplement
# d'économiser la saisie de quelques touches sur le clavier (à
# condition d'en saisir suffisamment pour éviter toute ambiguïté
# sur le nom du fichier).
SOURCE="/home/netlogon/clients-linux/dist*/squ*/int*/int*"

# Répertoire de destination sur le client GNU/Linux en local. Par
# exemple le bureau, histoire de voir apparaître le fichier
# sous nos yeux.
DESTINATION="/home/toto/Bureau/"

# Et enfin la copie du fichier du serveur vers le client GNU/Linux en local.
# Il faudra alors saisir le mot de passe du compte root du serveur.
scp root@IP-SERVEUR:"$SOURCE" "$DESTINATION"
```

Remarque : si jamais vous avez un doute sur le type de distribution de votre client GNU/Linux, vous pouvez lancer dans un terminal la commande suivante (pas forcément en tant que **root**) :

```
lsb_release --codename
```

Le résultat vous affichera le nom de code de la distribution (**squeeze** ou **precise** etc.) ce qui vous indiquera le script d'intégration à utiliser.

Supposons par exemple que vous avez copié le script d'intégration **integration_squeeze.bash** sur une Debian Squeeze et que celui-ci se trouve sur votre bureau. Alors, **en tant que root**, vous pouvez lancer l'intégration ainsi :

```
# D'abord, on se place sur le bureau (ici, il s'agit du bureau de toto).
cd /home/toto/Bureau

# Ensuite, on rend le script exécutable.
chmod u+x integration_squeeze.bash

# Enfin, on lance l'intégration.
./integration_squeeze.bash --nom-client="toto-04" --is --ivl --rc
```

Les explications sur les options se trouvent plus loin dans le document à la section 4 page 10. Si tout se passe bien, le client finira par lancer un redémarrage. Une fois celui-ci terminé, vous devriez être en mesure d'ouvrir une session avec un compte du domaine (comme le compte **admin** ou un compte de type professeur ou de type élève).

Il est préférable qu'aucun compte local du client n'ait le même login qu'un compte du domaine. Or, lorsqu'on installe un client GNU/Linux, on est en général amené à créer au moins un compte local (en plus du compte **root**). Si cela vous arrive, arrangez-vous pour que le login de ce compte ne risque pas de rentrer en conflit avec le login d'un compte du domaine. Vous pouvez utiliser **userlocal** comme login par exemple, ou autre chose...

3 Visite rapide du répertoire clients-linux/ du serveur

Afin de faire un rapide tour d'horizon du paquet **se3-clients-linux**, voici ci-dessous un schéma du contenu du répertoire **/home/netlogon/clients-linux/** du serveur. Les noms des répertoires possèdent un slash à la fin, sinon il s'agit de fichiers standards. Certains fichiers ou répertoires, dont vous n'avez pas à vous préoccuper, ont été omis afin d'alléger le schéma et les explications qui vont avec. Les fichiers ou répertoires que vous avez le droit de modifier pour les adapter à vos besoins sont en **vert**. À l'inverse, vous ne devez pas modifier tous les autres fichiers ou répertoires [1].

```
-- clients-linux/
   |-- bin/
   |   |-- connexion_ssh_serveur.bash
   |   |-- logon
   |   |-- logon_perso
   |   `-- reconfigure.bash
   |-- distribs/
   |   |-- precise/
   |   |   |-- integration/
   |   |   |   |-- desintegration_precise.bash
   |   |   |   `-- integration_precise.bash
   |   |   `-- skel/
   |   `-- squeeze/
   |       |-- integration/
   |       |   |-- desintegration_squeeze.bash
   |       |   `-- integration_squeeze.bash
   |       `-- skel/
   |-- divers/
   |-- doc/
   |   `-- LISEZMOI.TXT
   `-- unefois/
```

Voici quelques commentaires rapides :

- Le répertoire **bin/** contient en premier lieu le fichier **logon** qui est le script de logon. Ce script est véritablement le chef d'orchestre de tous les clients GNU/Linux intégrés au domaine. C'est lui qui contient les instructions exécutées systématiquement par les clients GNU/Linux juste avant l'affichage de la fenêtre de connexion, au moment de l'ouverture de session et au moment de la fermeture de session. Ce script de logon sera expliqué à la section 9 page 26. En principe, vous ne devez pas modifier ce fichier. En revanche, vous pourrez modifier le fichier **logon_perso** juste à côté. Ce fichier vous permettra d'affiner le comportement du script **logon** afin de l'adapter à vos besoins. Vous trouverez toutes les explications nécessaires dans la section 9.3 page 28.

  Le répertoire **bin/** contient également le fichier **connexion_ssh_serveur.bash**. Il s'agit simplement d'un petit script exécutable qui, lorsque sous serez connecté(e) avec le compte **admin** sur un client GNU/Linux et que vous double-cliquerez dessus, vous permettra d'ouvrir une connexion SSH sur le serveur en tant que **root** (autrement dit une console à distance sur le serveur en tant que **root**). C'est une simple commodité. Bien sûr, il vous sera demandé de fournir le mot de passe du compte **root** sur le serveur. Pour fermer proprement la connexion SSH, il vous suffira de taper sur la console la commande **exit**.

1. En fait, vous pouvez le faire bien sûr car vous êtes **root** sur le serveur. Mais les modifications effectuées sur les fichiers/répertoires qui ne sont pas en vert sur le schéma ne survivront pas à une réinstallation ou à une mise à jour du paquet **se3-clients-linux**.

Enfin, le répertoire **bin/** contient le fichier **reconfigure.bash**. Il s'agit d'un fichier exécutable très pratique qui vous permettra de remettre les droits par défaut sur l'ensemble des fichiers du paquet **se3-clients-linux** se trouvant sur le serveur et d'insérer le contenu du fichier **logon_perso** (votre fichier personnel que vous pouvez modifier afin d'ajuster le comportement des clients GNU/-Linux selon vos préférences) à l'intérieur du fichier **logon** qui est le seul fichier lu par les clients GNU/Linux. Vous pourrez lancer cet exécutable à partir du compte **admin** du domaine sur un client GNU/Linux intégré. Cet exécutable utilise une connexion SSH en tant que **root** et à chaque fois il faudra donc saisir le mot de passe **root** du serveur.

- Le répertoire **distribs/** contient un sous-répertoire par distribution GNU/Linux prise en charge par le paquet. Par exemple, dans le sous-répertoire **squeeze/**, il y a les dossiers suivants :
  - Un dossier **integration/** qui contient notamment le script d'intégration. C'est ce script qu'il faudra exécuter en tant que **root** sur chaque client Squeeze que l'on souhaite intégrer au domaine du serveur. Les options disponibles dans ce scripts sont décrites dans la section 4 à la page 10. Le script de « désintégration » se trouve également dans ce dossier, mais ce script est copié sur chaque client GNU/Linux en local au moment de l'intégration. Voir la section 5 page 13 pour plus d'explications sur le script de « désintégration ».
  - Un dossier **skel/** qui contient le profil par défaut (c'est-à-dire le home par défaut) de tous les utilisateurs du domaine sur la distribution concernée. Si vous voulez modifier la page d'accueil du navigateur de tous les utilisateurs du domaine ou bien si vous voulez ajouter des icônes sur le bureau, c'est dans ce dossier qu'il faudra faire des modifications. Vous trouverez toutes les explications nécessaires dans la section 7 à la page 16.
- Le répertoire **divers/** ne contient pas grand chose par défaut et vous pourrez a priori y mettre ce que vous voulez. L'intérêt de ce répertoire est que, si vous y placez des fichiers (ou des répertoires), ceux-ci seront accessibles uniquement par le compte **root** local de chaque client GNU/Linux et par le compte **admin** du domaine. En particulier, vous aurez accès au contenu du répertoire **divers/** à travers le script de logon et à travers les scripts « unefois » (évoqués ci-dessous) qui sont tous les deux exécutés par le compte **root** local de chaque client GNU/Linux. Vous trouverez un exemple d'utilisation possible de ce répertoire dans la section 12 à la page 41.
- Le répertoire **doc/** contient un fichier texte qui vous indiquera l'adresse URL de la documentation en ligne que vous êtes en train de lire actuellement (à savoir le fichier **SE3_clients_linux.pdf**) ainsi que l'adresse URL des sources au format LaTeX de cette documentation.
- Le répertoire **unefois/** sert à exécuter des scripts une seule fois sur toute une « famille » de clients GNU/Linux intégrés au domaine. Ce répertoire peut s'avérer utile pour effectuer des tâches administratives sur les clients GNU/Linux. Toutes les explications nécessaires sur ce répertoire se trouvent dans la section 8 page 20.

4 Les options des scripts d'intégration

Les deux scripts d'intégration **integration_squeeze.bash** et **integration_precise.bash**, qui doivent être exécutés en tant que **root** en local sur chaque client GNU/Linux à intégrer, utilisent exactement le même jeu d'options. En voici la liste.

- L'option **--nom-client** ou **--nc** : cette option vous permet de modifier le nom d'hôte[2] du client. Si l'option n'est pas spécifiée, alors le client gardera le nom d'hôte qu'il possède déjà. Si l'option est spécifiée sans paramètre, alors le script d'intégration stoppera son exécution pour vous demander de saisir le nom de la machine. Si l'option est spécifiée avec un paramètre, comme dans :

2. Celui qui se trouve dans le fichier **/etc/hostname**. Ce n'est pas un nom DNS pleinement qualifié.

```
./integration_squeeze.bash --nom-client="toto-04"
```

alors le script ne stoppera pas son exécution et effectuera directement le changement de nom en prenant comme nom le paramètre fourni (ici `toto-04`). Les caractères autorisés pour le choix du nom sont :

- les 26 lettres de l'alphabet en minuscules ou en majuscules, **sans accents** ;
- les chiffres ;
- le « tiret du 6 » (-) ;
- et c'est tout !

De plus, **le nom de la machine ne soit pas faire plus de 15 caractères**.

- L'option `--mdp-grub` ou `--mg` : cette option vous permet d'ajouter un mot de passe dès qu'un utilisateur souhaite éditer un des items du menu Grub au démarrage. En effet, en général, sur un système GNU/Linux fraîchement installé et utilisant Grub comme chargeur de boot, il est possible de sélectionner un des items du menu Grub et de l'éditer en appuyant sur la touche `e` sans devoir saisir le moindre mot de passe. Cela constitue une faille de sécurité potentielle car, dans ce cas, l'utilisateur peut très facilement éditer un des item du menu Grub et démarrer ensuite via cet item modifié de manière à devenir `root` sur la machine **sans avoir à saisir le moindre mot de passe**. Avec l'option `--mg`, quand l'utilisateur voudra éditer un des items du menu Gub, il devra saisir les identifiants suivants :
  - login : `admin` ;
  - mot de passe : celui spécifié avec l'option `--mg`.

  Si l'option `--mg` n'est pas spécifiée, alors la configuration de Grub est inchangée et a priori la faille de sécurité sera toujours présente. Si l'option est spécifié sans paramètre, alors le script d'intégration stoppera son exécution pour vous demander de saisir (deux fois) le futur mot de passe Grub (votre saisie ne s'affichera pas à l'écran). Si l'option est spécifiée avec un paramètre comme dans :

```
./integration_squeeze.bash --mdp-grub="1234"
```

  alors le script ne stoppera pas son exécution et effectuera directement le changement de configuration de Grub en prenant comme mot de passe le paramètre fourni (ici `1234`).

- L'option `--mdp-root` ou `--mr` : cette option vous permet de modifier le mot de passe du compte `root`. Si vous ne spécifiez pas cette option, le mot de passe du compte `root` sera inchangé. Si vous spécifiez cette option sans paramètre, alors le script d'intégration stoppera son exécution pour vous demander de saisir (deux fois) le futur mot de passe du compte `root` (votre saisie ne s'affichera pas sur l'écran). Si l'option est spécifiée avec un paramètre comme dans :

```
./integration_squeeze.bash --mdp-root="abcd"
```

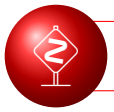
  alors le script ne stoppera pas son exécution et effectuera directement le changement de mot de passe en utilisant la valeur fournie en paramètre (ici `abcd`).

- L'option `--ignorer-verification-ldap` ou `--ivl` : cette option, qui ne prend aucun paramètre, vous permet de continuer l'intégration sans faire de pause après la vérification LDAP. En effet, lors de l'exécution du script d'intégration, quel que soit le jeu d'options choisi, une recherche dans l'annuaire du serveur est effectuée. Le script lancera une recherche de toutes les entrées dans l'annuaire correspondant à des machines susceptibles d'avoir un lien avec la machine qui est en train d'exécuter le script d'intégration au domaine. Plus précisément la recherche porte sur toutes les entrées dans l'annuaire correspondant à des machines qui ont :

- même nom que la machine exécutant le script ;
- **ou** même adresse IP que la carte réseau de la machine exécutant le script ;
- **ou** même adresse MAC que la carte réseau de la machine exécutant le script.

Dans tous les cas, le résultat de cette recherche sera affiché. Si vous n'avez pas spécifié l'option `--ivl`, alors le script s'arrêtera à ce moment là et vous demandera si vous voulez continuer l'intégration. Si par exemple vous vous apercevez que le nom d'hôte que vous avez choisi pour votre client GNU/Linux existe déjà dans l'annuaire du serveur, il faudra peut-être arrêter l'intégration (sauf si le système GNU/Linux est installé en dual boot avec Windows sur la machine et que le système Windows, lui, a déjà été intégré au domaine avec ce même nom). Mais si vous avez spécifié l'option `--ivl`, alors après avoir affiché le résultat de la recherche LDAP, le script continuera automatiquement l'intégration sans vous demander de confirmation.

- L'option `--installer-samba` ou `--is` : cette option, qui ne prend aucun paramètre, provoquera l'installation de Samba sur le client GNU/Linux. Si vous ne spécifiez pas cette option, alors Samba ne sera pas installé sur le client GNU/Linux. Actuellement, il est conseillé de spécifier cette option. En effet, lorsqu'un client GNU/Linux essaye de monter un partage Samba du serveur (notamment le partage `homes`), des scripts sont exécutés en amont côté serveur et le montage ne sera effectué qu'une fois ces scripts terminés. Or, l'un d'entre eux peut mettre un certain temps (environ 4 ou 5 secondes) à se terminer si Samba n'est pas installé sur la machine cliente. Par conséquent, si vous ne spécifiez pas l'option `--is`, vous risquez d'avoir des ouvertures de sessions un peu lentes (lors du montage des partages Samba). Donc pour l'instant, utilisez cette option lors de vos intégrations.



Pour l'instant, il faut utiliser l'option `--is` systématiquement.

- L'option `--redemarrer-client` ou `--rc` : cette option permet de lancer automatiquement un redémarrage du client GNU/Linux à la fin de l'exécution du script d'intégration. Si vous ne spécifiez pas cette option, il n'y aura pas de redémarrage à la fin de l'exécution du script. Sachez que le redémarrage après intégration est nécessaire pour avoir un système opérationnel. Si les intégrations se déroulent sans erreur sur vos machines Linux, vous aurez donc tout intérêt à spécifier à chaque fois l'option `--rc`.

Précisons enfin que, quel que soit le jeu d'options que vous aurez choisi, **aucun enregistrement dans l'annuaire du serveur ne sera effectué par le script d'intégration**. Par conséquent, si vous souhaitez que votre client GNU/Linux fraîchement intégré figure dans l'annuaire du serveur, il faudra passer par une réservation d'adresse IP de la carte réseau du client via le module DHCP du serveur.

Une fois un client intégré au domaine, évitez de monter un disque ou un partage dans le répertoire `/mnt/`. En effet, le répertoire `/mnt/` est utilisé constamment par le client GNU/Linux (une fois que celui-ci est intégré au domaine) pour y effectuer des montages de partages, notamment au moment de l'ouverture de session d'un utilisateur du domaine, et ce répertoire est aussi constamment « nettoyé », notamment juste après une fermeture de session. Afin d'éviter le « nettoyage » intempestif d'un de vos disques ou d'un partage réseau de votre cru, utilisez un autre répertoire pour procéder au montage. Utilisez par exemple le répertoire `/media/` à la place. En fait, utilisez ce que vous voulez sauf `/mnt/`.

5 La « désintégration »

Une fois un client GNU/Linux intégré au domaine, celui-ci possédera **localement** un script permettant de le faire « sortir » du domaine et de lui redonner (quasiment) son état avant l'intégration. Il s'agit du script :

`/etc/se3/bin/desintegration_<nom-de-code>.bash`

où vous pouvez remplacer `<nom-de-code>` par `squeeze`, par `precise` etc. Ce script admet une unique option (qui ne prend pas de paramètre) : il s'agit de l'option `--redemarrer-client` ou `--rc` qui, comme son nom l'indique, redémarre la machine à la fin du script de « désintégration ». Sans cette option, la machine ne redémarrera pas automatiquement. Tout comme pour les scripts d'intégration, après « désintégration », un redémarrage est nécessaire pour que le système soit opérationnel. Autre point commun : aucune modification sur l'annuaire du serveur n'est effectuée lors de l'exécution du script de « désintégration ». En particulier, après avoir « sorti » un client GNU/Linux du domaine, il faudra effacer vous-même toute trace de ce client dans l'annuaire du serveur.

6 Les partages des utilisateurs

6.1 Liste par défaut des partages accessibles suivant le type de compte

Attention, cette liste (décrite ci-dessous) est une liste proposée **par défaut** par le paquet. Vous verrez plus loin, à la section 9.5 page 31, que vous pourrez définir vous-même la liste des partages disponibles en fonction du compte qui se connecte, en fonction de son appartenance à tel ou tel groupe etc. **Cette liste est donc tout à fait modifiable**.

Avertissement valable uniquement pour ceux qui ont déjà installé une version n du paquet avec $n < 1.1$

Attention, depuis la version 1.1 du paquet, la gestion des partages accessibles se fait exclusivement dans le fichier **logon_perso**. Cela a une conséquence importante si une version antérieure à la version 1.1 du paquet est déjà installée sur votre serveur. En effet, lors de la mise à jour du paquet vers une version ≥ 1.1, plus aucun partage réseau ne devrait être monté à l'ouverture de session sur vos clients GNU/Linux et cela pour tout utilisateur du domaine.

C'est parfaitement normal car, lors de la mise à jour du paquet, votre fichier **logon_perso** a été conservé et c'est désormais dans ce fichier que les commandes de montage des partages sont effectuées. Or, a priori, votre fichier **logon_perso** ne contient pas encore ces commandes de montage.

Il est cependant très facile de retrouver le comportement par défaut (comme décrit ci-dessous) au niveau du montage des partages réseau à l'ouverture de session. Sur une console du serveur, en tant que **root**, il vous suffit de faire :

```
# On se place dans le répertoire bin/.
cd /home/netlogon/clients-linux/bin/

# On met dans un coin votre fichier logon_perso en le renommant
# logon_perso.SAVE (si jamais vous n'avez jamais touché à ce fichier
# alors vous pouvez même le supprimer avec la commande rm logon_perso).
mv logon_perso logon_perso.SAVE

# On reconfigure le paquet. L'absence du fichier logon_perso sera
# détectée et vous retrouverez ainsi la version par défaut de ce
# fichier.
dpkg-reconfigure se3-clients-linux
```

Vous retrouverez un comportement par défaut dès que les clients GNU/Linux auront mis à jour leur script de logon local, c'est-à-dire au plus tard après un redémarrage des clients (en fait, après une simple fermeture de session, la mise à jour devrait se produire).

Voici la liste, par défaut, des partages accessibles en fonction du type de compte lors d'une session.

1. **Un compte élève** aura accès :
   - Au partage **//SERVEUR/homes/Docs/** via deux liens symboliques. Tous les deux possèdent le même nom : « **Documents de <login> sur le réseau** ». L'un se trouve dans le répertoire **/home/<login>/** et l'autre dans le répertoire **/home/<login>/Bureau/**.
   - Au partage **//SERVEUR/Classes/** via deux liens symboliques. Tous les deux possèdent le même nom : « **Classes sur le réseau** ». L'un se trouve dans le répertoire **/home/<login>/** et l'autre dans le répertoire **/home/<login>/Bureau/**.
2. **Un compte professeur** aura accès :
   - Aux mêmes partages qu'un compte élève.
   - Mais il aura accès en plus au partage **//SERVEUR/Docs/** via deux liens symboliques. Tous les deux possèdent le même nom : « **Public sur le réseau** ». L'un se trouve dans le répertoire

`/home/<login>/` et l'autre dans le répertoire `/home/<login>/Bureau/`.

3. **Le compte `admin` aura accès :**
   - Aux mêmes partages qu'un compte professeur.
   - Mais il aura accès en plus au partage `//SERVEUR/admhomes/` via deux liens symboliques. Tous les deux possèdent le même nom : « `admhomes` ». L'un se trouve dans le répertoire `/home/admin/` et l'autre dans le répertoire `/home/admin/Bureau/`.
   - Et il aura accès en plus au partage `//SERVEUR/netlogon-linux/` via deux liens symboliques. Tous les deux possèdent le même nom : « `clients-linux` ». L'un se trouve dans le répertoire `/home/admin/` et l'autre dans le répertoire `/home/admin/Bureau/`.

6.2 Le lien symbolique clients-linux

Rien de nouveau donc au niveau des partages disponibles, à part le partage `netlogon-linux` accessible via le compte `admin` du domaine à travers le lien symbolique `clients-linux` situé sur le bureau. Ce lien symbolique vous permet d'avoir accès, en lecture et en écriture, au répertoire `/home/netlogon/clients-linux/` du serveur. Techniquement, une modification de ce répertoire est aussi possible via le lien symbolique `admhomes` puisque celui-ci donne accès à tout le répertoire `/home/` du serveur.

Avertissement : toujours reconfigurer les droits après modifications du contenu du répertoire clients-linux/

Lors de certains paramétrages du paquet `se3-clients-linux`, vous serez parfois amené(e) à modifier le contenu du répertoire `/home/netlogon/clients-linux/` du serveur :

- soit via une console sur le serveur si vous êtes un(e) adepte de la ligne de commandes ;
- soit via le lien symbolique `clients-linux` situé sur le bureau du compte `admin` lorsque vous est connecté(e) sur un client GNU/Linux intégré au domaine.

Dans un cas comme dans l'autre, une fois vos modifications terminées, il faudra **TOUJOURS** reconfigurer les droits du paquet `se3-clients-linux` sans quoi vous risquez ensuite de rencontrer des erreurs incompréhensibles. Pour ce faire il faudra :

- ou bien, **si vous êtes connecté(e) en mode console sur le serveur**, exécuter en tant que `root` la commande :

```
dpkg-reconfigure se3-clients-linux
```

- ou bien, **si vous êtes connecté(e) en tant qu'`admin` sur un client GNU/Linux**, double-cliquer sur le fichier `reconfigure.bash` accessible en passant par le lien symbolique `clients-linux` sur le bureau puis par le répertoire `bin/` (le mot de passe `root` du serveur sera demandé).

Remarque : en réalité, ces deux procédures ne font pas que reconfigurer les droits sur les fichiers, elles permettent aussi d'injecter le contenu du fichier `logon_perso` dans le fichier `logon`. Ce point sera abordé dans la section 9.3 page 28.

7 La gestion des profils

7.1 Une précision à avoir en tête

Dans cette documentation, on appellera « profil » le contenu **ou une copie du contenu** du home d'un utilisateur (par exemple le profil de `toto` est le contenu du répertoire `/home/toto/`). Pour bien comprendre le mécanisme des profils, il faut avoir en tête ces deux éléments :

1. Le serveur Samba offre un partage CIFS dont le chemin réseau est `//SERVEUR/netlogon-linux/` et qui correspond sur le serveur au répertoire `/home/netlogon/clients-linux/`.
2. Sur chaque client GNU/Linux intégré au domaine, le répertoire `/mnt/netlogon/` est un point de montage (en lecture seule) du partage CIFS `//SERVEUR/netlogon-linux/`.

Conclusion à bien avoir en tête : sur un client GNU/Linux intégré au domaine, visiter le répertoire local `/mnt/netlogon/` revient en fin de compte à visiter le répertoire `/home/netlogon/clients-linux/` du serveur Samba. Dans le tableau ci-dessous, les trois « adresses » suivantes désignent finalement la même zone de stockage qui se trouve sur le serveur :

Chemin réseau	Chemin local sur le serveur	Chemin local sur un client GNU/Linux
`//SE3/netlogon-linux`	`/home/netlogon/clients-linux/`	`/mnt/netlogon/`

7.2 Les différentes copies du profil par défaut

Revenons maintenant à nos profils. En fin de compte, pour un client GNU/Linux donné qui a été intégré au domaine, il existe plusieurs copies du profil par défaut des utilisateurs. **Dans le cas des clients sur Debian Squeeze** par exemple, il y a :

1. Le profil par défaut **distant** qui est unique et centralisé sur le serveur. Il est accessible de plusieurs manières. Les trois « adresses » ci-dessous accèdent toutes à ce même profil par défaut **distant** :
   - à travers le réseau via le partage CIFS :

     `//SERVEUR/netlogon-linux/distribs/squeeze/skel/`
   - sur le serveur directement à l'adresse :

     `/home/netlogon/clients-linux/distribs/squeeze/skel/`
   - sur chaque client Squeeze intégré au domaine via le chemin :

     `/mnt/netlogon/distribs/squeeze/skel/`
2. Le profil par défaut **local** qui se trouve sur chaque client intégré au domaine dans le **répertoire local** `/etc/se3/skel/` (ce répertoire n'est pas un point de montage, c'est un répertoire local au client GNU/Linux).

7.3 Le mécanisme des profils

Voici comment fonctionne le mécanisme des profils du point de vue d'un client GNU/Linux sous Debian Squeeze (sous une autre distribution, c'est exactement la même chose) :

1. **Au moment de l'affichage de la fenêtre de connexion** du système (c'est-à-dire soit juste après le démarrage du système ou soit juste après chaque fermeture de session), le client GNU/-Linux va comparer le contenu de deux fichiers :
   a. le fichier `/etc/se3/skel/.VERSION` de son profil par défaut **local**

b. le fichier `/mnt/netlogon/distribs/squeeze/skel/.VERSION` du profil par défaut **distant**.

Si ces deux fichiers ont un contenu totalement identique, alors le client GNU/Linux ne fait rien car il estime que son profil par défaut **local** et le profil par défaut **distant** sont identiques. Si en revanche les deux fichiers ont un contenu différent, alors le client va modifier son profil par défaut **local** afin qu'il soit identique au profil par défaut **distant**. Autrement dit, il va synchroniser [3] son profil par défaut **local** par rapport au profil par défaut **distant**.

2. **Au moment de l'ouverture de session** d'un compte du domaine, c'est-à-dire juste après une saisie correcte du login et du mot de passe d'un compte du domaine, appelons ce compte `toto`, le client GNU/Linux va créer le répertoire (local) vide `/home/toto/` et le remplir en y copiant dedans le contenu de son profil par défaut **local** (c'est-à-dire le contenu du répertoire `/etc/se3/skel/`) afin de compléter le home de `toto`.
3. **Au moment de la fermeture de session**, tous les liens symboliques situés dans `/home/toto/` qui permettent d'atteindre les différents partages auxquels `toto` peut prétendre sont supprimés.
4. **Au moment du prochain affichage de la fenêtre de connexion**, c'est-à-dire ou bien juste après la fermeture de session de `toto` s'il n'a pas choisi d'éteindre le poste client ou bien au prochain démarrage du système, **le répertoire `/home/toto/` est tout simplement effacé**.

7.4 Exemple de modification du profil par défaut avec Firefox

Du point de vue de l'utilisateur, cette gestion des profils est assez contraignante : par exemple notre cher `toto` aura beau modifier son profil durant sa session (changer le fond d'écran, ajouter un lanceur sur le bureau), après une fermeture puis réouverture de session, il retrouvera inlassablement le même profil par défaut et toutes ses modifications auront disparu. De plus, tous les comptes du domaine (que ce soit les comptes professeur ou les comptes élève) possèdent exactement le même profil par défaut [4]. Seule la liste des partages réseau accessibles sera différente d'un compte à l'autre. Mais ceci étant dit, cette gestion des profils présente tout de même deux avantages importants :

1. **Ouverture de session rapide :** en effet, au moment de l'ouverture de session d'un compte du domaine, la création du home ne sollicite pas le réseau puisqu'elle passe par une simple copie locale du contenu de `/etc/se3/skel/` qui est copié dans `/home/toto/`.
2. **Modification du profil par défaut (pour tous les utilisateurs) simple et rapide :** en effet, il devient très facile de modifier le profil par défaut des utilisateurs, car, si vous avez bien suivi, c'est le profil par défaut **distant** (celui sur le serveur) qui sert de modèle à tous les profils par défaut **locaux** des clients GNU/Linux. Une modification du profil par défaut **distant** accompagnée d'une modification du fichier `.VERSION` associé sera impactée sur chaque profil par défaut **local** de tous les clients GNU/Linux.

Prenons un exemple avec le navigateur Firefox : vous souhaitez imposer un profil par défaut particulier au niveau de Firefox pour tous les utilisateurs du domaine sur les clients GNU/Linux de type Precise Pangolin. Pour commencer, vous devez ouvrir une session sur un client GNU/Linux Precise Pangolin et lancer Firefox afin de le configurer exactement comme vous souhaitez qu'il le soit pour tous les utilisateurs (page d'accueil, proxy, etc). Une fois le paramétrage effectué, pensez bien sûr à fermer l'application Firefox. Ensuite, il vous suffit de suivre la procédure ci-dessous. Pour la suite, on admettra que la session utilisée pour fabriquer le profil Firefox par défaut est celle du compte `toto`.

3. Le terme de synchronisation est bien adapté car c'est justement la commande `rsync` qui est utilisée pour effectuer cette tâche.
4. Cette restriction pourra, dans une certaine mesure, être levée lorsqu'on abordera la personnalisation du script de logon à la section 9.3 page 28.

1. Il faut copier le répertoire **/home/toto/.mozilla/** [5] (et tout son contenu bien sûr) dans le profil par défaut **distant** du serveur, et cela tout en veillant à ce que les droits sur la copie soient corrects. Pour ce faire, vous avez deux méthodes possibles :
   - **Méthode graphique :** vous copiez le répertoire **/home/toto/.mozilla/** sous une clé USB puis vous fermez la session de **toto** pour en rouvrir une avec le compte **admin** du domaine. Ensuite, vous double-cliquez sur le lien symbolique **clients-linux** qui se trouve sur le bureau puis vous vous rendez successivement dans **distribs/** → **precise/** → **skel/** pour enfin, via un glisser-déposer, copier dans **skel/** le répertoire **.mozilla/** qui se trouve dans la clé USB (le dossier **skel/** devra donc contenir un répertoire **.mozilla/**).
     Attention, en général, les répertoires dont le nom commence par un point sont cachés par défaut et pour qu'ils s'affichent dans l'explorateur de fichiers il faudra sans doute activer une option du genre « **afficher les fichiers cachés** ».
     Enfin, comme vous avez ajouté des fichiers dans le répertoire **clients-linux/** du serveur, il faut reconfigurer les droits des fichiers. Pour ce faire, vous double-cliquez sur le lien symbolique **clients-linux** qui se trouve sur le bureau puis vous vous rendez dans **bin/** et vous double-cliquez sur le fichier **reconfigure.bash** (vous devrez saisir le mot de passe **root** du serveur).
   - **Méthode via la ligne de commandes :** sur la session de **toto** restée ouverte, vous ouvrez un terminal et vous lancez les commandes suivantes :

```
# Répertoire du client GNU/Linux à copier sur le serveur.
SOURCE="/home/toto/.mozilla/"

# Destination sur le serveur.
DESTINATION="/home/netlogon/clients-linux/distribs/precise/skel/"

# Copie du répertoire local (et de tout son contenu) vers le serveur.
scp -r "$SOURCE" root@IP-SERVEUR:"$DESTINATION"
```

     À ce stade, le répertoire **.mozilla/** a bien été copié sur le serveur mais les droits Unix sur la copie ne sont pas encore corrects. Pour les reconfigurer, il faut exécuter la commande « **dpkg-reconfigure se3-clients-linux** » en tant que **root** sur le serveur. Là aussi, cela peut se faire directement du client GNU/Linux, sans bouger, via ssh avec la commande :

```
# Avec ssh, en étant sur le client GNU/Linux, on peut exécuter notre commande
# à distance sur le serveur tant que root.
ssh -t root@IP-SERVEUR "dpkg-reconfigure se3-clients-linux"
```

2. Modifiez le fichier **.VERSION** [6] du profil par défaut **distant**. Ce fichier **.VERSION** est un simple fichier texte, vous pouvez le modifier avec un simple éditeur. S'il contient la chaîne « 1 » par exemple, alors éditez-le et écrivez « 2 » à la place. Si vous préférez, vous pouvez très bien indiquer la date du moment comme dans « Le 13 décembre 2013 à 15h04 ». Le but est simplement, qu'une fois modifié, le fichier **.VERSION** du serveur possède un contenu différent de chacun des fichiers **.VERSION** locaux aux machines clientes. Dans notre exemple, le fichier se trouve dans le répertoire **/home/netlogon/clients-linux/precise/skel/** du serveur. Là aussi, deux méthodes s'offrent à vous pour le modifier :
   - **La méthode graphique** : si ce n'est pas déjà fait, vous fermez la session de **toto** pour vous connecter sur le client GNU/Linux avec le compte **admin** du domaine. Ensuite, vous

5. Car c'est ce répertoire qui contient tous les réglages concernant Firefox que vous avez effectués.
6. Ne pas oublier cette étape, sans quoi les clients GNU/Linux estimeront que le profil par défaut **distant** n'a pas été modifié et la mise à jour du profil par défaut **local** n'aura pas lieu

double-cliquez sur le lien symbolique **clients-linux** qui se trouve sur le bureau puis vous vous rendez successivement dans **distribs/** → **precise/** → **skel/**. Faites en sorte d'activer l'option « **afficher les fichiers cachés** » afin de voir apparaître le fichier **.VERSION** qui se trouve à l'intérieur du dossier **skel/**. Éditez ce fichier afin simplement de modifier son contenu. Bien sûr, pensez à enregistrer la modification. Pas besoin ici de reconfigurer les droits car le fait de modifier le contenu du fichier **.VERSION** ne change pas les droits sur ce fichier qui, a priori, étaient déjà corrects.

- **Méthode via la ligne de commandes :** sur la session de **toto** restée ouverte, vous ouvrez un terminal et vous lancez les commandes suivantes :

```
# Le fichier sur le serveur qu'il faut modifier.
CIBLE="/home/netlogon/clients-linux/distribs/precise/skel/.VERSION"

ssh root@IP-SERVEUR "echo Version du 10 janvier 2012 à 15h04 > $CIBLE"
# Maintenant le fichier contient "Version du 10 janvier 2012 à 15h04".
```

Dès le prochain affichage de la fenêtre de connexion, les profils par défaut **locaux** de tous les clients Precise Pangolin seront modifiés afin d'être identiques au profil par défaut **distant** du serveur. Dès lors, les utilisateurs bénéficieront des paramétrages de Firefox que vous avez effectués.

De la même manière que précédemment, sur le profil par défaut **distant**, vous pouvez parfaitement définir le contenu du bureau des utilisateurs : au lieu de copier un répertoire **.mozilla/** sur le serveur, ce sera un répertoire **Bureau/**, mais le principe reste le même.

D'une distribution à une autre, les versions des logiciels n'étant pas forcément identiques, chaque distribution prise en charge possède son propre profil par défaut **distant**. Sur le serveur Samba, on a donc :

- le répertoire **/home/netlogon/clients-linux/distribs/squeeze/skel/** pour les Debian Squeeze.
- le répertoire **/home/netlogon/clients-linux/distribs/precise/skel/** pour les Ubuntu Precise Pangolin.

7.5 Personnaliser le profil en fonction de l'utilisateur

La rigidité de la gestion du profil telle qu'elle est décrite à la section 7.4 peut cependant être contournée en modifiant le script de logon [7]. Pour comprendre cela, poursuivons avec l'exemple de la modification du profil par défaut de Mozilla. Imaginons que vous souhaitiez que les enseignants disposent d'un navigateur dont la configuration diffère de celle à laquelle accèdent les élèves (extensions particulières, favoris différents, etc.).

Dans ce cas, vous copierez sur le répertoire **/skel** du serveur le répertoire **/home/toto/.mozilla** après l'avoir renommé en **/.mozilla-prof**.

Évidemment, dans ce cas, si un enseignant ouvre une session et lance son navigateur, la configuration prise en compte par le système sera toujours celle du répertoire **/.mozilla**. Il faut donc, pour achever ce processus, modifier le fichier **logon_perso** pour qu'au moment de l'ouverture de session, le répertoire **/.mozilla** soit remplacé par **/.mozilla-prof** si et seulement si c'est un(e) enseignant(e) qui se connecte.

Pour ce faire, vous utiliserez les variables prêtes à l'emploi (voir section 9.4), et indiquerez dans la fonction **ouverture_perso** les lignes suivantes :

7. Le fonctionnement du script de logon est décrit dans la section 9, page 26.

```
 # chargement du profil mozilla pour les profs
if est_dans_liste "$LISTE_GROUPES_LOGIN" "Profs"; then
    rm -rf "$REP_HOME/.mozilla"
    mv "$REP_HOME/.mozilla-prof" "$REP_HOME/.mozilla"
fi
```

Ainsi, à l'ouverture de session, si l'utilisateur qui se connecte est un(e) enseignant(e), le **logon_perso** commencera par supprimer le répertoire **.mozilla**, puis renommera **/.mozilla-prof** en **/.mozilla**, permettant ainsi au système de prendre en compte ce répertoire pour la configuration du navigateur.

Vous imaginez la suite : on peut, avec cette méthode, personnaliser la configuration de tous les logiciels et de l'environnemnet de bureau pour chaque profil, voire pour chaque utilisateur.

8 Le répertoire unefois/

8.1 Principe de base

Si vous souhaitez faire des interventions ponctuelles sur les clients GNU/Linux sans vous déplacer devant les postes, alors le répertoire **/home/netlogon/clients-linux/unefois/** du serveur Samba peut vous intéresser. En effet, des fichiers exécutables placés dans ce répertoire seront susceptibles d'être lancés une seule fois sur les clients GNU/Linux lors du démarrage. En pratique, vous allez créer un sous-répertoire à la racine du répertoire **unefois/** du serveur. Par exemple :

- Si le nom de ce sous-répertoire est **dell740**, alors les exécutables se trouvant dans ce sous-répertoire seront lancés une fois au démarrage de tous les clients GNU/Linux dont le nom de machine **contient à la casse près** la chaîne de caractères **dell740**.
- Si le nom de ce sous-répertoire est **^S121-**[8], alors les exécutables se trouvant dans ce sous-répertoire seront lancés une fois au démarrage de tous les clients GNU/Linux dont le nom de machine **commence à la casse près par** la chaîne de caractères **S121-**.
- Si le nom de ce sous-répertoire est **prof$**, alors les exécutables se trouvant dans ce sous-répertoire seront lancés une fois au démarrage de tous les clients GNU/Linux dont le nom de machine **se termine à la casse près par** la chaîne de caractères **prof**.
- Si le nom de ce sous-répertoire est **^S121-HP-P$**, alors les exécutables se trouvant dans ce sous-répertoire seront lancés une fois au démarrage du client GNU/Linux dont le nom de machine **est identique à la casse près à** la chaîne de caractères **S121-HP-P**.

Si jamais cela évoque quelque chose pour vous, sachez qu'en réalité le nom des sous-répertoires est interprété par le client GNU/Linux comme une expression régulière étendue. Vous pouvez donc choisir comme nom de sous-répertoire n'importe quelle expression régulière étendue pour filtrer les noms de machines qui sont censées exécuter une fois vos scripts ou vos fichiers binaires.

Voici un dernier exemple de nom de sous-répertoire possible (et donc d'expression régulière possible) : **^.** (le nom de ce sous-répertoire est constitué d'un accent circonflexe puis d'un point). Cette expression régulière signifie : « n'importe quelle chaîne de caractères qui commence par un caractère quelconque ». Autrement dit, les exécutables se trouvant dans ce sous-répertoire seront lancés une fois au démarrage de **tous les clients GNU/Linux sans exception**. Bien sûr, le répertoire **unefois/** du serveur peut parfaitement contenir plusieurs sous-répertoires. Dans ce cas, si le nom de machine d'un client correspond par exemple avec trois noms de sous-répertoires **regex1/**, **regex2/** et **regex3/**, alors le client devra lancer une seule fois au démarrage tous les exécutables contenus dans chacun des sous-répertoires **regex1/**, **regex2/** et **regex3/**.

8. Oui, il s'agit bien d'un répertoire dont le nom commence par un accent circonflexe.

Après avoir créé vos sous-répertoires et vos fichiers exécutables dans le répertoire `unefois/` du serveur, n'oubliez pas de réajuster les droits sur les fichiers comme expliqué à la section 6.2 page 15.

Attention, les fichiers exécutables d'un sous-répertoire donné doivent vérifier certains critères :

- Le nom d'un exécutable **ne doit pas commencer par un point**.
- Le nom d'un exécutable **doit se terminer par** `.unefois` (comme dans `mon-script.unefois`).
- Si le fichier exécutable est un script (autrement dit si ce n'est pas un fichier binaire), **il doit impérativement comporter un shebang**[9] : cela peut-être un script Bash, Perl, Python peu importe (du moment que l'interpréteur du langage est installé sur les clients GNU/Linux) mais il faut que le shebang soit présent.

Le critère pour que les clients GNU/Linux se souviennent d'avoir exécuté un fichier donné (afin de l'exécuter une seule fois) est le nom de ce fichier et rien que le nom (pas le contenu). Par exemple, si un client GNU/Linux a exécuté le script `toto.unefois`, alors ce client n'exécutera plus jamais[10] de fichier s'appellant `toto.unefois`. Si vous avez un script que vous souhaitez exécuter non pas une seule fois, mais quelques fois de manière très ponctuelle (une fois par an par exemple), pensez à insérer la date du jour dans le nom du script (comme dans `1sept2012-maj.unefois`) et le cas échéant, en modifiant la date dans le nom du fichier (par exemple en le renommant `3sept2013-maj.unefois`), celui-ci sera à nouveau candidat à l'exécution du côté des clients GNU/Linux.

8.2 Le mécanisme en détail

Voici le mécanisme effectué par les clients GNU/Linux au niveau du répertoire `unefois/` **au moment du démarrage du système** uniquement (le démarrage est le seul instant où les clients GNU/Linux se préoccupent du répertoire `unefois/`) :

1. Le client regarde le contenu de tous les sous-répertoires de `/mnt/netlogon/unefois/`[11] dont les noms correspondent à son nom de machine. Par exemple, si le client s'appelle `S18-DELL-03`, il va regarder le contenu du sous-répertoire `^S18-` mais il va ignorer le sous-répertoire `-HP-`. Dans chaque sous-répertoire qu'il n'a pas ignoré (s'il en existe), le client va y chercher tous les fichiers de la forme `*.unefois`, afin d'obtenir toute une liste (éventuellement vide) de fichier `*.unefois`.
2. Si, dans cette liste de fichiers `*.unefois`, certains noms figurent déjà dans le répertoire local `/etc/se3/unefois/`, c'est que les fichiers en question ont déjà été exécutés par le client GNU/Linux et ils ne le sont donc pas une deuxième fois. En revanche, les fichiers de cette liste dont le nom[12] ne figure pas dans `/etc/se3/unefois/` sont copiés dans ce répertoire local puis les copies locales sont exécutées.

C'est donc le répertoire `/etc/se3/unefois/` qui constitue la « mémoire » du client GNU/Linux : il contient la liste des noms de fichiers déjà exécutés. Il y a toutefois deux exceptions au mécanisme décrit ci-dessus :

9. Le shebang est la première ligne d'un script qui commence par `#!` comme dans « `#! /bin/bash` » ou dans « `#! /usr/bin/python` ».
10. En fait, comme vous allez le voir juste après, cette règle n'est pas complètement immuable.
11. Rappelons à nouveau que le répertoire `/mnt/netlogon/unefois/` sur les clients GNU/Linux correspond en réalité au répertoire `/home/netlogon/clients-linux/unefois/` du serveur Samba.
12. Les clients GNU/Linux ne tiennent compte que du nom des fichiers, pas de leur contenu. La casse dans le nom des fichiers est prise en compte.

1. Au moment du démarrage du système, si le client détecte la présence d'un fichier nommé **PAUSE**[13] à la racine du répertoire **/mnt/netlogon/unefois/**, alors le client ne fait strictement rien au niveau des fichiers ***.unefois** et donc il n'exécute absolument rien, quoi qu'il arrive.
2. Au moment du démarrage du système, si le client ne repère pas la présence du fichier **PAUSE** précédent mais qu'en revanche il détecte la présence du fichier **BLACKOUT**[14], toujours à la racine du répertoire local **/mnt/netlogon/unefois/**, alors le client GNU/Linux efface le contenu du répertoire **/etc/se3/unefois/**. Ainsi, au prochain démarrage, si les fichiers **PAUSE** et **BLACKOUT** ne sont pas présents, le client exécutera tous les exécutables ***.unefois** qui le concerne, peu importe leur nom étant donné que la « mémoire » du client GNU/Linux concernant tout ce qui a déjà été exécuté a été effacée.

Au moment du démarrage, la recherche par les clients GNU/Linux des fichiers ***.unefois** à exécuter (ainsi que leur copie en local le cas échéant) entraîne(nt) forcément du trafic réseau. Lorsque vous ne souhaitez pas faire usage de ce mécanisme (ce qui en principe sera le cas 90% du temps), n'hésitez pas à placer le fichier **PAUSE** à la racine du répertoire **unefois/** du serveur afin d'éviter ce travail de recherche aux clients GNU/Linux qui solliciteraient inutilement le réseau.

Là encore, lorsque vous créerez ce fichier **PAUSE**, attention de bien reconfigurer les droits des fichiers comme expliqué à section 6.2 page 15.

Les scripts ***.unefois** sont tous exécutés, en tant que **root**, **en arrière-plan** et cela dès l'affichage de la fenêtre de connexion lors du démarrage. Si vous souhaitez qu'un script ***.unefois** se lance un peu après (parce que, par exemple, vous avez besoin d'attendre que certains services soient lancés), vous pouvez parfaitement utiliser des instructions comme « **sleep 20** » afin de forcer le script à attendre pendant 20 secondes avant de commencer réellement son travail. Enfin sachez que dans le répertoire local **/etc/se3/unefois/**, chaque exécutable **truc.unefois** est accompagné de son homologue nommé **truc.unefois.log** qui contient simplement l'ensemble des messages (d'erreur ou non) du fichier l'exécutable.

8.3 Réglage de la locale durant l'exécution des scripts « unefois »

Avant de déployer un script bash via le répertoire **unefois/** du serveur, il sera sans doute nécessaire de le tester sur un client localement. Sachez que les scripts bash, lorsqu'ils sont exécutés par le client GNU/Linux au démarrage, ont la variable d'environnement **LC_ALL** définie comme étant égale à **C**[15] et non pas égale à **fr_FR.utf8**. Cela implique que tous les messages de sortie des commandes système lancées dans le script seront en anglais avec des caractères ASCII uniquement. Pour avoir une idée de l'influence de la locale sur les commandes système, vous pouvez ouvrir un terminal bash et tester ceci :

```
# On paramètre le terminal sur une locale française qui doit être très
# probablement la locale par défaut déjà définie sur votre système.
export LC_ALL="fr_FR.utf8"
# Puis on teste une commande. En principe, l'entête du résultat de la
# commande est en français.
df -h
```

13. Le nom du fichier doit être en majuscules uniquement et peu importe le contenu de ce fichier qui peut être totalement vide. Attention, les droits de ce fichier doivent être corrects une fois celui-ci créé.
14. Même remarque que pour le fichier **PAUSE**.
15. Cette valeur règle le système, le temps de l'exécution des scripts, sur la locale standard **C** qui est la seule locale parfaitement normalisée et a priori disponible sur n'importe quel système de type Unix.

```
# Maintenant, on paramètre le terminal sur la locale C.
export LC_ALL="C"

# Et on teste à nouveau la même commande. Cette fois-ci, l'entête du
# résultat de la commande est en anglais.
df -h
```

Par conséquent, si jamais vous souhaitez exploiter le résultat de certaines commandes système dans vos scripts bash **`*.unefois`**, sachez que la locale peut avoir une incidence sur le comportement du script. Si jamais vous tenez à avoir une locale française lors de l'exécution de votre script, alors il vous suffit de placer juste en dessous du shebang l'instruction :

```
export LC_ALL="fr_FR.utf8"
```

En revanche, si vous ne souhaitez pas forcer le réglage sur une locale particulière et préférez conserver la valeur par défaut (avec la locale standard **`C`**), alors durant vos tests afin de valider un script bash à déployer, il faudra le lancer de la manière suivante :

```
LC_ALL="C" ./monscript.bash.unefois
```

De cette manière, le script héritera de la locale **`C`** et il se comportera de la même manière que lors d'une exécution via le mécanisme « unefois ». Alors que si vous lancez le script ainsi :

```
./monscript.bash.unefois
```

celui-ci hériterait de la locale du système, qui est très probablement **`fr_FR.utf8`**, et il se comporterait légèrement différemment que lors d'une exécution via la mécanisme « unefois », si bien que vos tests seraient légèrement biaisés.

8.4 Des variables et des fonctions prêtes à l'emploi

Si jamais vous utilisez le langage Bash pour écrire des script de la forme **`*.unefois`**, vous pouvez alors utiliser certaines variables ou fonctions prédéfinies qui pourront peut-être vous faciliter le travail d'écriture des scripts. Voici tableau listant toutes ces variables et fonctions :

Nom	Commentaire
`SE3`	Cette variable stocke l'adresse IP du serveur récupérée automatiquement lors de l'installation du paquet **`se3-clients-linux`**.
`NOM_DE_CODE`	Cette variable stocke ce qu'on appelle le « nom de code » de la distribution (**`squeeze`** dans le cas d'une Debian Squeeze, **`precise`** dans le cas d'une Ubuntu Precise Pangolin etc).
`ARCHITECTURE`	Cette variable stocke l'architecture du système. Par exemple, si le système repose sur une architecture 64 bits, alors la variable stockera la chaîne de caractères **`x86_64`**.
`BASE_DN`	Cette variable contient le suffixe de base LDAP de l'annuaire du serveur. Elle pourra vous être utile si vous souhaitez faire vous-même des requêtes LDAP particulières sur les clients à l'aide de la commande **`ldapsearch`**.

TABLE 1 – Variables et fonctions disponibles dans les scripts « unefois ».

Nom	Commentaire

NOM_HOTE	Cette variable stocke le nom du client GNU/Linux (celui qui se trouve dans le fichier de configuration /etc/hostname). Par exemple, si vous avez pris l'habitude de choisir des noms de machines de la forme <salle>-xxx (comme dans S121-PC04 ou même comme dans S18-DELL-02), alors vous pourrez récupérer le nom de la salle où se trouve le client GNU/Linux par l'intermédiaire de la variable NOM_HOTE comme ceci :
	`SALLE=$(echo "$NOM_HOTE" | cut -d'-' -f1)`
	
	`if [ "$SALLE" = "S121" ]; then`
	`  # Les trucs à faire si on est dans la salle 121.`
	`fi`
	
	`if [ "$SALLE" = "S18" ]; then`
	`  # Les trucs à faire si on est dans la salle 18.`
	`fi`
	`# etc.`

appartient_au_parc	Cette fonction permet de savoir si une machine appartient à un parc donné. Pour ce faire, la fonction appartient_au_parc interroge l'annuaire du serveur via une requête LDAP. Voici un exemple d'utilisation :
	`if appartient_au_parc "S121" "$NOM_HOTE"; then`
	`  # La machine appartient au parc S121`
	`else`
	`  # La machine n'appartient pas au parc S121`
	`fi`

afficher_liste_parcs	Un exemple vaudra mieux qu'un long discours : `liste_parcs=$(afficher_liste_parcs "S121-LS-P")` Dans cet exemple, la fonction effectue une requête LDAP auprès du serveur afin de connaître le nom de tous les parcs auxquels appartient la machine S121-LS-P. Si la machine S121-LS-P appartient aux parcs S121 et PostesProfs, alors la variable liste_parcs contiendra deux lignes, la première contenant S121 et la deuxième contenant PostesProfs. L'idée est de stocker tous les parcs d'une machine dans une variable, le tout en une seule requête LDAP. Enfin, à la place de S121-LS-P comme argument de la fonction, on aurait pu utiliser $NOM_HOTE, comme dans l'exemple ci-dessous qui sera plus éclairant sur la manière dont on peut exploiter de telles listes.

TABLE 1 – Variables et fonctions disponibles dans les scripts « unefois ».

<table>
<tr><th>Nom</th><th>Commentaire</th></tr>
<tr><td>est_dans_liste</td><td>Là aussi, illustrons cette fonction par un exemple :
<pre>
# On récupère la liste des parcs auxquels
# appartient la machine cliente.
liste_parcs=$(afficher_liste_parcs "$NOM_HOTE")

if est_dans_liste "$liste_parcs" "PostesProfs"; then
    # Si la machine est dans le parc "PostesProfs"
    # alors faire ceci...
elif est_dans_liste "$liste_parcs" "CDI"; then
    # Si la machine est dans le parc "CDI"
    # alors faire ceci...
else
    # Sinon faire cela...
fi
</pre>L'idée ici est qu'une seule requête LDAP est effectuée (lors de la première instruction). Ensuite, les tests if ne sollicitent pas le réseau puisque la liste des parcs est déjà stockée dans la variable liste_parcs.</td></tr>
<tr><td colspan="2">Les fonctions suivantes sont moins pertinentes dans les scripts *.unefois qui, rappelons-le, sont exécutés juste après le démarrage du système. Mais elles restent toutefois disponibles également et donc figurent quand même dans ce tableau. En revanche, nous verrons plus loin (à la section 9.4 page 29) que ces fonctions sont également disponibles à des moments beaucoup plus pertinents, comme par exemple au moment de l'ouverture de session d'un utilisateur sur le système.</td></tr>
<tr><td>appartient_au_groupe</td><td>Cette fonction permet de savoir si le login d'un utilisateur correspond à un compte qui appartient à un groupe donné. Pour ce faire, la fonction appartient_au_groupe interroge l'annuaire du serveur via une requête LDAP. Voici un exemple :
<pre>
if appartient_au_groupe "Classe_1ES2" "toto"; then
   # Le compte toto appartient à la classe 1ES2.
else
   # Le compte toto n'appartient pas à la classe 1ES2.
fi
</pre></td></tr>
</table>

TABLE 1 – Variables et fonctions disponibles dans les scripts « unefois ».

Nom	Commentaire

`afficher_liste_groupes`	Un exemple vaudra mieux qu'un long discours :
	`liste_groupes_toto=$(afficher_liste_groupes "toto")`
	`if est_dans_liste "$liste_groupes_toto" "Eleves"; then`
	`    # toto est un élève alors faire ceci...`
	`fi`
	Dans cet exemple, la fonction effectue une requête LDAP auprès du serveur afin de connaître le nom des groupes auxquels compte utilisateur `toto` appartient. Si par exemple ce compte appartient aux groupes `Eleves` et `Classe_1ES2`, alors la variable `liste_groupes_toto` contiendra deux lignes, la première contenant `Eleves` et la deuxième contenant `Classe_1ES2`. L'idée est de stocker tous les groupes d'un compte donné dans une variable, le tout en une seule requête LDAP.

`est_utilisateur_local`	Cette fonction permet de tester si un compte est local (c'est-à-dire contenu dans le fichier `/etc/passwd` du client GNU/Linux) ou non (c'est-à-dire un compte du domaine contenu dans l'annuaire du serveur).
	`if est_utilisateur_local "toto"; then`
	`    # toto est un compte local, alors faire ceci...`
	`fi`

`est_connecte`	Cette fonction permet de tester si un compte est actuellement connecté au système (c'est-à-dire s'il a ouvert une session).
	`if est_connecte "toto"; then`
	`    # toto est actuellement connecté au système,`
	`    # alors faire ceci...`
	`fi`

`activer_pave_numerique`	Cette fonction, qui ne prend pas d'argument, permet simplement d'activer le pavé numérique du client GNU/Linux.

TABLE 1 – Variables et fonctions disponibles dans les scripts « unefois ».

9 Le script de logon

9.1 Phases d'exécution du script de logon

Le script de logon est un script bash qui est exécuté par les clients GNU/Linux lors de trois phases différentes. Pour plus de commodité dans les explications, nous allons donner un nom à chacune de ces trois phases une bonne fois pour toutes :

1. **L'initialisation :** cette phase se produit juste avant l'affichage de la fenêtre de connexion. Attention, cela correspond en particulier au démarrage du système, certes, mais pas seulement. L'initialisation se produit aussi juste après la fermeture de session d'un utilisateur, avant que la

fenêtre de connexion n'apparaisse à nouveau (sauf si, bien sûr, l'utilisateur a choisi d'éteindre ou de redémarrer la machine).

Description rapide des tâches exécutées par le script lors de cette phase : le script efface les homes (s'il en existe) de tous utilisateurs qui ne correspondent pas à des comptes locaux[16], vérifie si le partage CIFS `//SERVEUR/netlgon-linux` du serveur est bien monté sur le répertoire `/mnt/netlogon/` du client GNU/Linux et, si ce n'est pas le cas, le script exécute ce montage. Ensuite, le cas échéant, le script procède à la synchronisation du profil par défaut local sur le profil par défaut distant et lance les exécutions des `*.unefois` si l'initialisation correspond en fait à un redémarrage du système.

2. **L'ouverture :** cette phase se produit à l'ouverture de session d'un utilisateur juste après que celui-ci ait saisi ses identifiants.

   Description rapide des tâches exécutées par le script lors de cette phase : le script procède à la création du home de l'utilisateur qui se connecte (via une copie du profil par défaut local), exécute le montage de certains partages du serveur auxquels l'utilisateur peut prétendre (comme par exemple le partage correspondant aux données personnelles de l'utilisateur).

3. **La fermeture :** cette phase se produit à la fermeture de session d'un utilisateur.

   Description rapide des tâches exécutées par le script lors de cette phase : le script ne fait rien qui mérite d'être signalé dans cette documentation.

Comme vous pouvez le constater, le script de logon est un peu le « chef d'orchestre » de chacun des clients GNU/Linux.

9.2 Emplacement du script de logon

À la base, le script de logon se trouve localement à l'adresse `/etc/se3/bin/logon` de chaque client GNU/Linux. Mais il existe une version centralisée de ce script sur le serveur à l'adresse :

1. `/home/netlogon/clients-linux/bin/logon` si on est sur le serveur
2. `/mnt/netlogon/bin/logon` si on est sur un client GNU/Linux

Nous avons donc, comme pour le profil par défaut, des versions locales du script de logon (sur chaque client GNU/Linux) et une unique version distante (sur le serveur). Et au niveau de la synchronisation, les choses fonctionnent de manière très similaire aux profils par défaut. **Lors de l'initialisation d'un client GNU/Linux** :

- Si le contenu du script de logon local est identique au contenu du script de logon distant, alors c'est le script de logon local qui est exécuté par le client GNU/Linux.
- Si en revanche les contenus diffèrent (ne serait-ce que d'un seul caractère), alors c'est le script de logon distant qui est exécuté. Mais dans la foulée, le script de logon local est écrasé puis remplacé par une copie de la version distante. Du coup, il est très probable qu'à la prochaine initialisation du client GNU/Linux ce soit à nouveau le script de logon local qui soit exécuté parce que identique à la version distante (on retombe dans le cas précédent).

A priori, cela signifie donc que, pour peu que vous sachiez parler (et écrire) le langage du script de logon (il s'agit du Bash), vous pouvez modifier **uniquement** le script de logon distant (celui du serveur donc) afin de l'adapter à vos besoins. Vos modifications seraient alors impactées sur **tous les clients** GNU/Linux dès la prochaine phase d'initialisation. Seulement, **il ne faudra pas procéder ainsi** et cela pour une raison simple : après la moindre mise à jour du paquet `se3-clients-linux` ou éventuellement après une réinstallation, toutes vos modifications sur le script de logon seront effacées. Pour pouvoir modifier le comportement du script de logon de manière pérenne, il faudra utiliser le fichier `logon_perso` qui se trouve dans le même répertoire que le script de logon.

16. Un compte local est un compte figurant dans le fichier `/etc/passwd` du client GNU/Linux.

9.3 Personnaliser le script de logon

Le fichier `logon_perso` va vous permettre d'affiner le comportement du script de logon afin de l'adapter à vos besoins, et cela de manière pérenne dans le temps (les modifications persisteront notamment après une mise à jour du paquet `se3-clients-linux`). À la base, le fichier `logon_perso` est un fichier texte encodé en UTF-8 avec des fins de ligne de type Unix [17]. Il contient du code bash et possède, par défaut, la structure suivante :

```
function initialisation_perso ()
{
   # ...
}

function ouverture_perso ()
{
   # ...
}

function fermeture_perso ()
{
   # ...
}
```

Ce code sera ni plus ni moins inclus, tel quel, dans le script de logon. En fait, après une modification du fichier `logon_perso`, il faudra donc **toujours** penser à reconfigurer le paquet `se3-clients-linux` comme décrit dans la section 6.2 page 15 ce qui aura pour effet, entre autres, d'insérer le contenu de la nouvelle version de `logon_perso` dans le fichier `logon`. Si vous oubliez de faire cette manipulation après modification du fichier `logon_perso`, le fichier `logon` sera inchangé et vos modifications ne seront tout simplement pas prises en compte.

Procédure à suivre quand on modifie le fichier `logon_perso`

Pour modifier le script de logon afin de l'adapter à vos besoins, vous devez :

1. Modifier le fichier `logon_perso`.
2. Puis lancer la reconfiguration du paquet `se3-clients-linux` en effectuant une des deux procédures décrites dans la section 6.2 page 15, afin que le contenu de la nouvelle version de `logon_perso` soit inséré dans le fichier `logon`.

Revenons au contenu du fichier `logon_perso` pour comprendre de quelle manière il permet de modifier le comportement du script `logon`. Dans le fichier `logon_perso`, on peut distinguer trois fonctions :

1. Tout le code que vous mettrez dans la fonction `initialisation_perso` sera exécuté lors de la phase d'initialisation des clients, **en dernier**, c'est-à-dire après que le script de logon ait effectué toutes les tâches liées à la phase d'initialisation qui sont décrites brièvement au point 1 de la section 9.1.

17. Attention d'utiliser un éditeur de texte respectueux de l'encodage et des fins de ligne lorsque vous modifierez le fichier `logon_perso`.

2. Tout le code que vous mettrez dans la fonction `ouverture_perso` sera exécuté lors de la phase d'ouverture des clients uniquement lorsqu'un utilisateur du domaine se connecte. Le code est exécuté **juste après** la création du « home » de l'utilisateur qui se connecte. Typiquement, c'est dans cette fonction que vous allez gérer les montages de partages réseau en fonction du type de compte qui se connecte (son appartenance à tel ou tel groupe etc).

   Pour la gestion des montages de partages réseau à l'ouverture de session, tout se trouve à la section 9.5 page 31.

3. Tout le code que vous mettrez dans la fonction `fermeture_perso` sera exécuté lors de la phase de fermeture des clients, **en dernier**, c'est-à-dire après que le script de logon ait effectué toutes les tâches liées à la phase de fermeture qui sont décrites brièvement au point 3 de la section 9.1.

Vous pouvez bien sûr définir dans le fichier `logon_perso` des fonctions supplémentaires, mais, pour que celles-ci soient au bout du compte exécutées par le script de logon, il faudra les appeler dans le corps d'une des trois fonctions `initialisation_perso`, `ouverture_perso` ou `fermeture_perso`.

Il faut bien avoir en tête que le contenu de `logon_perso` est ni plus ni moins inséré dans le script `logon` et donc, après modification de `logon_perso`, il faut toujours mettre à jour le fichier `logon` via la commande « `dpkg-reconfigure se3-clients-linux` ».

9.4 Quelques variables et fonctions prêtes à l'emploi

Voici la liste des variables et des fonctions que vous pourrez utiliser dans le fichier `logon_perso` et qui seront susceptibles de vous aider à affiner le comportement du script de logon :

Nom	Commentaire
Pour commencer, toutes les variables et les fonctions présentées dans le tableau 1 à la page 26 sont utilisables.	
`LOGIN`	Cette variable stocke le login de l'utilisateur qui a ouvert une session. Cette variable n'a de sens que lors de la phase d'ouverture et de fermeture (c'est-à-dire uniquement à l'intérieur des fonctions `ouverture_perso` et `fermeture_perso`), pas lors de la phase d'initialisation (c'est-à-dire à l'intérieur de la fonction `initialisation_perso`) puisque personne n'a encore ouvert de session à ce moment là.
`NOM_COMPLET_LOGIN`	Cette variable stocke le nom complet (sous la forme « prénom nom ») de l'utilisateur qui a ouvert une session. Cette variable n'a de sens que lors de la phase d'ouverture et de fermeture.
`REP_HOME`	Cette variable stocke le chemin absolu du répertoire home de l'utilisateur qui se connecte. Par exemple, si le compte `toto` ouvre une session, la variable contiendra la chaîne `/home/toto`. Remarquez que cette variable est un simple raccourci pour écrire `"/home/$LOGIN"`. Cette variable n'a de sens que lors de la phase d'ouverture et de fermeture.

TABLE 2 – Variables et fonctions disponibles dans le fichier logon_perso.

Nom	Commentaire

LISTE_GROUPES_LOGIN	Cette variable, qui n'a de sens **que lors de la phase d'ouverture**, stocke la liste des groupes auxquels appartient l'utilisateur qui a ouvert une session (le format étant un nom de groupe par ligne). Une utilisation typique de cette variable est :
	`if est_dans_liste "$LISTE_GROUPES_LOGIN" "Profs"; then`
	`  # L'utilisateur qui se connecte appartient au`
	`  # groupe Profs, alors faire ceci...`
	`elif est_dans_liste "$LISTE_GROUPES_LOGIN" "Eleves"; then`
	`  # L'utilisateur qui se connecte appartient au`
	`  # groupe Eleves, alors faire cela...`
	`fi`
	Au passage, dans ce code, aucune requête LDAP n'est effectuée puisque la variable LISTE_GROUPES_LOGIN contient déjà la liste des groupes auxquels appartient l'utilisateur qui vient de se connecter (la requête LDAP permettant de définir la variable LISTE_GROUPES_LOGIN a été faite par le script de logon en amont, une fois pour toute).

DEMARRAGE	Cette variable stocke toujours la valeur false, sauf lorsqu'on se trouve lors d'une phase d'initialisation qui correspond à un démarrage du système où elle stocke alors la valeur true. Cette variable n'a donc d'intérêt que lorsqu'elle est utilisée dans la fonction initialisation_perso. Voici un exemple :
	`if "$DEMARRAGE"; then`
	`  # On est lors d'une phase de démarrage`
	`  # alors faire ceci...`
	`fi`

monter_partage	Si vous voulez que les utilisateurs du domaine puissent avoir accès à des partages réseau sur le serveur, il faudra forcément faire usage de cette fonction qui est donc très importante. Toutes les explications sur cette fonction se trouvent à la section 9.5 page 31. Cette fonction n'a de sens que lors de la phase d'ouverture.
creer_lien	Cette fonction, qui va de pair avec la précédente, sera détaillée à la section 9.5 page 31. Cette fonction n'a de sens que lors de la phase d'ouverture.
changer_icone	Cette fonction sera détaillée à la section 9.6.1 page 35. Cette fonction n'a de sens que lors de la phase d'ouverture.
changer_papier_peint	Cette fonction, utilisable uniquement pendant la phase d'ouverture, permet de changer le fond d'écran de l'utilisateur qui se connecte. Elle prend un argument qui correspond au chemin absolu (sur le client) du fichier image à utiliser en guise de fond d'écran. Un exemple de l'utilisation de cette fonction sera donné à la section 9.6.2 page 36.
activer_pave_numerique	Cette fonction, qui fait exactement ce à quoi on pense naturellement, sera détaillée à la section 9.6.3 page 37.

TABLE 2 – Variables et fonctions disponibles dans le fichier logon_perso.

Nom	Commentaire
executer_a_la_fin	Parfois certaines commandes nécessitent d'être exécutées un fois le script de logon terminé (c'est-à-dire une fois l'initialisation, l'ouverture ou la fermeture terminée). C'est ce que permet cette fonction. Avec par exemple : `executer_a_la_fin "5" "commande" "arg1" "arg2"` la commande commande (avec ses arguments) sera lancée 5 secondes après que le script de logon ait terminé son exécution. Un exemple de l'usage de cette fonction sera donné à la section 9.6.4 page 38. Attention, l'exécution se faisant une fois le script de logon terminé, il y aura aucune trace dans les fichiers de log de l'exécution de la commande commande.

TABLE 2 – Variables et fonctions disponibles dans le fichier logon_perso.

9.5 Gestion du montage des partages réseau

Comme cela a déjà été expliqué, c'est vous qui allez gérer les montages de partages réseau en éditant le contenu de la fonction ouverture_perso qui se trouve dans le fichier logon_perso. Évidemment, si la gestion par défaut des montages vous convient telle quelle, alors vous n'avez pas besoin de toucher à ce fichier. Commençons par un exemple simple :

```
function ouverture_perso ()
{
    monter_partage "//$SE3/Classes" "Classes" "$REP_HOME/Bureau/Répertoire Classes"
}
```

Ici la fonction monter_partage possède trois **arguments qui devront être délimités par des doubles quotes** (") :

1. Le premier représente le chemin UNC du partage à monter. Vous reconnaissez sans doute la variable SE3 qui stocke l'adresse IP du serveur. Par exemple si l'adresse IP du serveur est 172.20.0.2, alors le premier argument sera automatiquement développé en :

   //172.20.0.2/Classes.

   Cela signifie que c'est le partage Classes du serveur 172.20.0.2 qui va être monté sur le clients GNU/Linux. Attention, sous GNU/Linux un chemin UNC de partage s'écrit avec des slashs (/) et non avec des antislashs (\) comme c'est le cas sous Windows.

2. Maintenant, il faut un répertoire local pour monter un partage. C'est le rôle du deuxième argument. Quoi qu'il arrive (vous n'avez pas le choix sur ce point), le partage sera monté dans un sous-répertoire du répertoire /mnt/_$LOGIN/. Par exemple si c'est toto qui se connecte sur le poste client, le montage sera fait dans un sous répertoire de /mnt/_toto/. Le deuxième argument spécifie le nom de ce sous-répertoire. Ici nous avons décidé assez logiquement de l'appeler Classes. Par conséquent, en visitant le répertoire /mnt/_toto/Classes/ sur le poste client, notre cher toto aura accès au contenu du partage Classes du serveur.

   Attention, dans le choix du nom de ce sous-répertoire, vous êtes limité(e) aux **caractères a-z, A-Z, 0-9, le tiret (-) et le tiret bas (_)**. C'est tout. En particulier **pas d'espace ni accent**. Si vous ne respectez pas cette consigne le partage ne sera tout simplement pas monté et une fenêtre d'erreur s'affichera à l'ouverture de session.

Vous serez sans doute amené(e) à monter plusieurs partages réseau pour un même utilisateur (via plusieurs appels de la fonction `monter_partage` au sein de la fonction `ouverture_perso`). Donc il y aura plusieurs sous-répertoires dans `/mnt/_$LOGIN/`. Charge à vous d'éviter les doublons dans les noms des sous-répertoires, sans quoi certains partages ne seront pas montés.

3. À ce stade, notre cher `toto` pourra accéder au partage `Classes` du serveur en passant par `/mnt/_toto/Classes/`. Mais cela n'est pas très pratique. L'idéal serait d'avoir accès à ce partage directement via un dossier sur le bureau de `toto`. C'est exactement ce que fait le troisième argument. Si `toto` ouvre une session, l'argument `"$REP_HOME/Bureau/Répertoire Classes"` va se développer en `"/home/toto/Bureau/Répertoire Classes"` si bien qu'un raccourci (sous GNU/Linux on appelle ça un lien symbolique) portant le nom `Répertoire Classes` sera créé sur le bureau de `toto`. Donc en double-cliquant sur ce raccourci (vous pouvez voir à la page 35 via une capture d'écran que ce genre de raccourci ressemble à un simple dossier), sans même le savoir, `toto` visitera le répertoire `/mnt/_toto/Classes/` qui correspondra au contenu du partage `Classes` du serveur. Vous n'êtes pas limité(e) dans le choix du nom de ce raccourci. Les espaces et les accents sont parfaitement autorisés (évitez par contre le caractère double-quote). En revanche, ce raccourci doit forcément être créé dans le home de l'utilisateur qui se connecte. **Donc ce troisième argument devra toujours commencer par** `"$REP_HOME/..."` sans quoi le lien ne sera tout simplement pas créé.

Tout n'a pas encore été dévoilé concernant cette fonction `monter_partage`. En fait, vous pouvez créer autant de raccourcis que vous voulez. Il suffit pour cela d'ajouter un quatrième argument, puis un cinquième , puis un sixième etc. Voici un exemple :

```
function ouverture_perso ()
{
    monter_partage "//$SE3/Classes" "Classes" \<Touche ENTRÉE>
        "$REP_HOME/Bureau/Lecteur réseau Classes" \<Touche ENTRÉE>
        "$REP_HOME/Lecteur réseau Classes"
}
```

> **Remarque :** normalement il faut mettre une fonction avec ses arguments sur une même ligne car un saut de ligne signifie la fin d'une instruction aux yeux de l'interpréteur Bash. Mais ici la ligne serait bien longue à écrire et dépasserait la largeur de la page de ce document. La combinaison antislash (\) puis ENTRÉE permet simplement de passer à la ligne tout en signifiant à l'interpréteur Bash que l'instruction entamée n'est pas terminée et qu'elle se prolonge sur la ligne suivante.

Le premier argument correspond toujours au chemin UNC du partage réseau et le deuxième argument au nom du sous-répertoire dans `/mnt/_$LOGIN/` associé à ce partage. Ensuite, nous avons cette fois-ci un troisième **et un quatrième argument** qui correspondent aux raccourcis pointant vers le partage : l'un est créé sur le bureau et l'autre est créé à la racine du home de l'utilisateur qui se connecte. Il est possible de créer autant de raccourcis que l'on souhaite, il suffit d'empiler les arguments 3, 4, 5 etc. les uns à la suite des autres.

La syntaxe de la fonction `monter_partage` est donc la suivante :

```
monter_partage "<partage>" "<répertoire>" ["<raccourci>"]...
```

où seuls les deux premiers arguments sont obligatoires :

- `<partage>` est le chemin UNC du partage à monter. Il est possible de se limiter à un sous-répertoire du partage, par exemple comme dans `//$SE3//administration/docs` où l'on montera uniquement le sous-répertoire `docs/` du partage `administration` du serveur.
- `<répertoire>` est le nom du sous-répertoire de `/mnt/_$LOGIN/` qui sera créé et sur lequel le partage sera monté. Seuls les caractères `-_a-zA-Z0-9` sont autorisés.

- Les arguments **<raccourci>** sont optionnels. Ils représentent les chemins absolus des raccourcis qui seront créés et qui pointeront vers le partage. Ils doivent toujours se situer dans le home de l'utilisateur qui se connecte, donc ils doivent toujours commencer par **"$REP_HOME/..."**. Si ces arguments ne sont pas présents, alors le partage sera monté mais aucun raccourci ne sera créé.

Attention, le montage du partage réseau se fait avec les droits de l'utilisateur qui est en train de se connecter. Si l'utilisateur n'a pas les droits suffisants pour accéder à ce partage, ce dernier ne sera tout simplement pas monté.

Remarque : au final, si vous placez bien vos raccourcis, l'utilisateur n'aura que faire du répertoire **"/mnt/_$LOGIN/"**. Il utilisera uniquement les raccourcis qui se trouvent dans son home. Peu importe pour lui de savoir qu'ils pointent en réalité vers un sous-répertoire de **"/mnt/_$LOGIN/"**, il n'a pas à s'en préoccuper.

Remarque : je vous conseille de toujours créer au moins un raccourci à la racine du home de l'utilisateur qui se connecte. En effet, lorsqu'un utilisateur souhaite enregistrer un fichier via une application quelconque, très souvent l'explorateur de fichiers s'ouvre au départ à la racine de son home. C'est donc un endroit privilégié pour placer les raccourcis vers les partages réseau. Il me semble que doubler les raccourcis à la fois à la racine du home et sur le bureau de l'utilisateur est une bonne chose. Mais bien sûr, tout cela est une question de goût...

Étant donné que le montage d'un partage se fait avec les droits de l'utilisateur qui se connecte, certains partages devront être montés uniquement dans certains cas. Prenons l'exemple du partage **netlogon-linux** du serveur. Celui-ci n'est accessible qu'au compte **admin** du domaine. Pour pouvoir monter ce partage seulement quand c'est le compte **admin** qui se connecte, il va falloir ajouter ce bout de code dans la fonction **ouverture_perso** du fichier **logon_perso** :

```
function ouverture_perso ()
{
    # Montage du partage "netlogon-linux" seulement dans le cas
    # où c'est le compte "admin" qui se connecte.
    if [ "$LOGIN" = "admin" ]; then
        # Cette partie là ne sera exécutée qui si c'est admin qui se connecte.
        monter_partage "//$SE3/netlogon-linux" "clients-linux" \<Touche ENTRÉE>
            "$REP_HOME/clients-linux" \<Touche ENTRÉE>
            "$REP_HOME/Bureau/clients-linux"
    fi
}
```

Remarque : attention, en Bash, le crochet ouvrant au niveau du **if** doit absolument être précédé et suivi d'un espace et le crochet fermant doit absolument être précédé d'un espace.

Autre cas très classique, celui d'un partage **accessible uniquement à un groupe**. Là aussi, une structure avec un **if** s'impose :

```
function ouverture_perso ()
{
    # On décide que le montage du partage "administration" sera seulement effectué si
    # c'est un compte qui appartient au groupe "Profs" qui se connecte.
    if est_dans_liste "$LISTE_GROUPES_LOGIN" "Profs"; then
        monter_partage "//$SE3/administration" "administration" \<Touche ENTRÉE>
            "$REP_HOME/administration sur le réseau" \<Touche ENTRÉE>
```

```
            "$REP_HOME/Bureau/administration sur le réseau"
    fi
}
```

L'instruction « `if est_dans_liste "$LISTE_GROUPES_LOGIN" "Profs"; then` » doit s'interpréter ainsi : « si dans la liste des groupes dont est membre le compte qui se connecte actuellement il y a le groupe **Profs**, autrement dit si le compte qui se connecte actuellement appartient au groupe **Profs**, alors... »

Attention, le test **if** ci-dessus est sensible à la casse si bien que le résultat ne sera pas le même si vous mettez **"Profs"** ou **"profs"**. Par conséquent, prenez bien la peine de regarder le nom du groupe qui vous intéresse avant de l'insérer dans un test **if** comme ci-dessus afin de bien respecter les minuscules et les majuscules.

Si vous voulez savoir le nom des partages disponibles pour un utilisateur donné, par exemple **toto**, il vous suffit de lancer la commande suivante sur le serveur en tant que **root** :

```
smbclient --list localhost -U toto
# Il faudra alors saisir le mot de passe de toto.
```

Parmi la liste des partages, l'un d'eux est affiché sous le nom de **home**. Il correspond au home de **toto** sur le serveur. Ce partage est un peu particulier car il pointera vers un répertoire différent en fonction du compte qui tente d'y accéder. Par exemple, si **titi** veut accéder à ce partage, alors il sera rédirigé vers le répertoire **/home/titi/** du serveur. Chaque utilisateur a le droit de monter ce partage, mais attention le chemin UNC est en fait **//SERVEUR/homes** (avec un « s » à la fin et d'ailleurs dans le fichier de configuration Samba ce partage est bien défini par la section **homes**). A priori, on pourra monter ce partage pour tous les comptes du domaine donc pas besoin de structure **if** pour ce partage :

```
function ouverture_perso ()
{
    # Montage du sous-répertoire "Docs" du partage "homes" pour tout le monde.
    monter_partage "//$SE3/homes/Docs" "Docs" \<Touche ENTRÉE>
        "$REP_HOME/Documents de $LOGIN sur le réseau" \<Touche ENTRÉE>
        "$REP_HOME/Bureau/Documents de $LOGIN sur le réseau"
}
```

Dans l'exemple ci-dessus, on ne monte pas le partage **homes** mais uniquement le sous-répertoire **Docs** de ce partage. Comme d'habitude sous GNU/Linux, respectez bien la casse des noms de partages et de répertoires.

Pour l'instant, de par la manière dont la fonction **monter_partage** est définie, on peut créer uniquement des liens qui pointent vers la racine du partage associé. Mais on peut vouloir par exemple monter un partage et créer des liens uniquement vers des sous-répertoires de ce partage (et non vers sa racine). C'est tout à fait possible avec la fonction **creer_lien**. Voici un exemple :

```
function ouverture_perso ()
{
    # Montage du partage "homes" pour tout le monde, mais ici on ne créé pas de
    # lien vers la racine de ce partage (appel de la fonction avec seulement deux
    # arguments).
    monter_partage "//$SE3/homes" "home"

    # Ensuite on crée des liens mais ceux-ci ne pointent pas à la racine du partage.
```

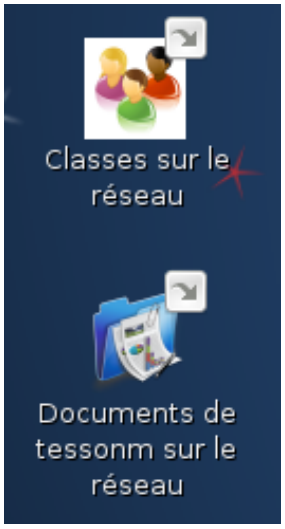
```
    creer_lien "home/Docs" "$REP_HOME/Documents de $LOGIN sur le réseau"
    creer_lien "home/Bureau" "$REP_HOME/Bureau de $LOGIN sous Windows"
}
```

Le premier argument de la fonction `creer_lien` est la cible du ou des liens à créer. Cette cible peut s'écrire sous la forme d'un chemin absolu, c'est-a-dire un chemin qui commence par un antislash (ce qui n'est pas le cas ci-dessus). Si le chemin ne commence pas par un antislash, alors la fonction part du principe que c'est un chemin relatif qui part de `/mnt/_$LOGIN/`[18]. Ensuite, le deuxième argument et les suivants (autant qu'on veut) sont les chemins absolus du ou des liens qui seront créés. Ces chemins doivent impérativement tous commencer par `"$REP_HOME/..."`.

9.6 Quelques bricoles pour les perfectionnistes

9.6.1 Changer les icônes représentants les liens pour faire plus joli

C'est quand même plus joli quand on a des icônes évocateurs[19] comme ci-dessous pour nos liens vers les partages, non ?



Et bien ça tombe bien car c'est facile à faire avec la fonction `changer_icone`. Voici un exemple :

```
function ouverture_perso ()
{
    # On suppose que le partage "Classe" est déjà monté et qu'un
    # lien vers ce partage a déjà été créé sur le bureau...
    changer_icone "$REP_HOME/Bureau/Classes sur le réseau" \<Touche ENTRÉE>
                  "$REP_HOME/.mes_icones/classe.jpg"
}
```

La fonction prend toujours deux arguments. Le premier est le chemin absolu du fichier dont on veut changer l'icône. Cela peut être n'importe quel fichier (ce n'est pas forcément un des raccourcis qu'on a créé), mais par contre il doit impérativement se trouver dans le home de l'utilisateur qui se connecte (donc il devra toujours commencer par `"$REP_HOME/..."`). Ensuite, le deuxième argument est le chemin absolu de n'importe quel fichier image (du moment que le compte qui se connecte peut y avoir accès en lecture).

18. Du coup, mettre `"home/Docs"` ou mettre `/mnt/_$LOGIN/home/Docs` comme premier argument revient exactement au même.
19. En informatique, le masculin est autorisé pour le mot icône.

Une idée possible (parmi d'autres) est de modifier le profil par défaut des d'utilisateurs et d'y placer un répertoire `.mes_icones/` dans lequel vous mettez tous les icônes dont vous avez besoin pour habiller vos liens. Ensuite, vous pourrez aller chercher vos icônes dans le home de l'utilisateur qui se connecte (dans `"$REP_HOME/.mes_icones/"` précisément) de manière similaire à ce qui est fait dans exemple ci-dessus.

Attention, la fonction `changer_icone` n'a aucun effet sous la distribution Xubuntu qui utilise l'environnement de bureau Xfce. Cela vient du fait que personnellement je ne sais pas changer l'image d'un icône en ligne de commandes sous Xfce. Si vous savez, n'hésitez pas à me donner l'information par mail car je pourrais ainsi étendre la fonction `changer_icone` à l'environnement de bureau Xfce.

9.6.2 Changer le papier peint en fonction des utilisateurs

Ça pourrait être sympathique d'avoir un papier différent suivant le type de compte... Et bien c'est possible avec la fonction `changer_papier_peint`. Voici un exemple :

```
function ouverture_perso ()
{
    if [ "$LOGIN" = "admin" ]; then
        changer_papier_peint "$REP_HOME/.backgrounds/admin.jpg"
    fi
}
```

Le seul et unique argument de cette fonction est le chemin absolu (sur la machine cliente) du fichier image servant pour le fond d'écran. Il faut bien sûr que ce fichier image soit au moins accessible en lecture pour l'utilisateur qui se connecte.

Là aussi, comme pour les icônes, l'idée est de placer dans le profil par défaut distant un répertoire `.backgrounds/` (par exemple) qui contiendra les deux ou trois fichiers images dont vous avez besoin pour faire vos fonds d'écran. Voici un exemple dans le cas d'un compte professeur :

En plus du changement de fond d'écran, il y a un petit message personnalisé qui s'affiche en haut à droite du bureau. Pour mettre en place ce genre de message, voir la section 9.6.4 page 38.

9.6.3 L'activation du pavé numérique

Pour activer le pavé numérique du client GNU/Linux au moment de l'affichage de la fenêtre de connexion du système, en principe ceci devrait fonctionner :

```
function initialisation_perso ()
{
    # On active le pavé numérique au moment de la phase d'initialisation.
    activer_pave_numerique
}
```

Vous pouvez remarquer que, cette fois-ci, c'est le contenu de la fonction `initialisation_perso` qui a été édité.

En revanche, pour activer le pavé numérique au moment de l'ouverture de session, procéder exactement de la même façon à l'intérieur de la fonction `ouverture_perso` risque de ne pas fonctionner, et cela pour une raison de timing. En effet, au moment où la fonction `ouverture_perso` sera lancée, l'ouverture de session ne sera pas complètement terminée [20] et l'activation du pavé numérique risque d'être annulée lors de la fin de l'ouverture de session. L'idée est donc de programmer l'appel de la fonction `activer_pave_numerique` **après** l'exécution du script de logon, seulement au bout de quelques secondes (par exemple 5), afin de lancer l'activation du pavé numérique une fois l'ouverture de session achevée :

20. Et c'est normal qu'il en soit ainsi puisque l'ouverture de session de termine **après** l'exécution du script de logon, même pas immédiatement après mais 1 ou 2 secondes après selon la rapidité de la machine hôte.

```
function ouverture_perso ()
{
    # On ajoute un argument à l'appel de la fonction activer_pave_numerique.
    # Ici, cela signifie que l'activation du pavé numérique sera lancée 5
    # secondes après que le script de logon soit terminé, ce qui laissera
    # le temps à l'ouverture de session de se terminer.
    activer_pave_numerique "5"
}
```

9.6.4 Incruster un message sur le bureau des utilisateurs pour faire classe

Pour incruster un message sur le bureau des utilisateurs, il faudra d'abord que le paquet **conky** soit installé [21] sur le client GNU/Linux. Ensuite, tentez de mettre ceci dans la fonction **ouverture_perso** :

```
function ouverture_perso ()
{
    # On crée un fichier de configuration .conkyrc dans le home de l'utilisateur.
    # précisant le contenu du message ainsi que certains paramètres (comme la
    # taille de la police par exemple).
    cat > "$REP_HOME/.conkyrc" <<FIN
use_xft yes
xftfont Arial:size=10
double_buffer yes
alignment top_right
update_interval 1
own_window yes
own_window_transparent yes
override_utf8_locale yes
text_buffer_size 1024
own_window_hints undecorated,below,sticky,skip_taskbar,skip_pager
TEXT
Bonjour $NOM_COMPLET_LOGIN,
Pensez bien à enregistrer vos données personnelles
dans le dossier :

     Documents de $LOGIN sur le réseau

qui se trouve sur le bureau, et uniquement dans ce
dossier, sans quoi vos données seront perdues une
fois votre session fermée.

   Cordialement.
   Les administrateurs du réseau pédagogique.
FIN

    # On fait de "$LOGIN" le propriétaire du fichier .conkyrc.
    chown "$LOGIN:" "$REP_HOME/.conkyrc"
```

21. Vous pouvez par exemple lancer l'installation via un script ***.unefois** qui contiendrait à peu de choses près l'instruction **apt-get install --yes conky**.

```
    chmod 644 "$REP_HOME/.conkyrc"

    # On lancera conky à la fin, une fois l'exécution du script logon terminée.
    # Pour être sûr que l'ouverture de session est achevée, on laisse un délai
    # de 5 secondes entre la fin du script de logon et le lancement de la
    # commande conky (avec ses arguments).
    executer_a_la_fin "5" conky --config "$REP_HOME/.conkyrc"
}
```

En principe, vous devriez voir apparaître un message incrusté sur le bureau des utilisateurs en haut à droite. Ce message sera légèrement personnalisé puisqu'il contiendra le nom de l'utilisateur connecté.

9.6.5 Exécuter des commandes au démarrage tous les 30 jours

Toutes les commandes que vous mettrez à l'intérieur de la fonction **initialisation_perso** du fichier **logon_perso** seront exécutées à chaque phase d'initialisation du système ce qui peut parfois s'avérer un peu trop fréquent à votre goût. Voici un exemple de fonction **initialisation_perso** qui vous permettra d'exécuter des commandes (peu importe lesquelles ici) au démarrage du système tous les 30 jours (pour peu que le système ne reste pas éteint indéfiniment bien sûr) :

```
function initialisation_perso ()
{
    local indicateur
    indicateur="/etc/se3/action_truc"
    # Si le fichier n'existe pas alors il faut le créer.
    [ ! -e "$indicateur" ] && touch "$indicateur"

    # On teste si la phase d'initialisation correspond à un démarrage du système.
    if "$DEMARRAGE"; then
        # On teste si la date de dernière modification du fichier est > 29 jours.
        if find "$indicateur" -mtime +29 | grep -q "^$indicateur$"; then
            echo "Les conditions sont vérifiées, on lance les actions souhaitées."
            action1
            action2
            # etc.

            # Si tout s'est bien déroulé, alors on peut mettre à jour la date
            # de dernière modification du fichier avec la commande touch.
            if [ "$?" = "0"  ]; then
                touch "$indicateur"
            fi
        fi
    fi
}
```

L'idée de ce code est plus simple qu'il n'y paraît. Chaque client GNU/Linux intégré au domaine possède un répertoire local **/etc/se3/** (accessible en lecture et en écriture au compte **root** uniquement). Dans ce répertoire, le script y place un fichier texte vide qui se nomme **action_truc** (c'est un exemple) et dont le seul but est de fournir une date de dernière modification. Au départ, cette date de dernière modification coïncide au moment où le fichier est créé. Si, lors d'un prochain démarrage, cette date de dernière modification est vieille de 30 jours ou plus, alors les actions sont exécutées et la date de

dernière modification du fichier `action_truc` est modifiée artificiellement en la date du jour avec la commande `touch`.

10 Les logs pour détecter un problème

Après modification du script de logon, vous n'obtiendrez peut-être pas le comportement souhaité. Peut-être parce que vous aurez tout simplement commis des erreurs. Afin de faire un diagnostic, il vous sera toujours possible de consulter, **sur un client GNU/Linux**, quelques fichiers log qui se trouvent tous dans le répertoire `/etc/se3/log/`. Voici la liste des fichiers log disponibles :

- `0.maj_logon.log` : la mise à jour du script de logon local (via son remplacement par une copie de la version distante) est un moment important et ce fichier indiquera si cette mise à jour a marché ou non. La date de la mise à jour y est indiquée.
- `1.initialisation.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution du script de logon **local** lors de la phase d'initialisation.
- `1.initialisation_distant.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution, lors de la phase d'initialisation, du script de logon **distant** (celui qui se trouve sur le serveur) et non celui qui se trouve en local sur le client GNU/Linux. Rappelez-vous que cela se produit quand les deux versions du script de logon (la version locale et la version et distante) sont différentes (ce qui est censé se produire ponctuellement seulement puisque la version locale est ensuite mise à jour).
- `1.initialisation_perso.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution, lors de la phase d'initialisation, de votre fonction `initialisation_perso`.
- `2.ouverture.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution du script de logon local lors de la phase d'ouverture.
- `2.ouverture_perso.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution, lors de la phase d'ouverture, de votre fonction `ouverture_perso`.
- `3.fermeture.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution du script de logon local lors de la phase de fermeture.
- `3.fermeture_perso.log` : ce fichier contiendra tous les messages (d'erreur ou non) suite à l'exécution, lors de la phase de fermeture, de votre fonction `fermeture_perso`.

À chaque fois que le script de logon s'exécute, avant d'écrire sur le fichier `xxx.log` adapté à la situation du moment, le fichier `xxx.log`, s'il existe déjà, est d'abord vidé de son contenu. Donc les fichiers log ne seront jamais très gros. Par exemple, dans le fichier `1.initialisation.log`, vous aurez des informations portant uniquement sur la dernière phase d'initialisation effectuée par le client GNU/Linux (pas sur les phases d'initialisation précédentes).

11 Le cas des classes nomades

Utiliser GNU/Linux sur des ordinateurs portables dans un domaine Se3 présente un atout extraordinaire : le mécanisme des profils (voir section 7.3, page 16) limite au maximum les échanges entre le serveur et le client une fois la session ouverte. Autrement dit, il n'y a aucun risque de voir la session ouverte « planter » en raison d'une micro-coupure wifi.

L'intégration au domaine d'un ordinateur issu d'une classe nomade ne présente qu'une spécificité : le client doit, avant l'ouverture de session, déjà être connecté au réseau sans fil. Pour ce faire, il suffira d'indiquer dans le fichier `/etc/network/interfaces` le SSID et le mot de passe du réseau sans fil auquel le client est censé se connecter.. Il est également recommandé, par la même occasion, de

désactiver l'interface ethernet, sans quoi le processus de boot se trouvera allongé de plusieurs secondes voire dizaines de secondes (durant lesquelles le client cherchera à obtenir une IP du serveur sur toutes les interfaces activées).

Un moyen extrêmement simple et rapide de réaliser cette manipulation est bien sûr d'utiliser **unefois/**. Ainsi, admettons que les clients de votre classe nomade soit nommés **nomade-01**, **nomade-02**, jusqu'à **nomade-15**. Avant leur intégration au domaine, vous pouvez par exemple :

- déposer dans le répertoire **/var/www/** de votre Se3 le fichier **interfaces** configuré par vos soins
- préparer un script intitulé **reseau-nomade.unefois** contenant les lignes suivantes [22] :

```
#!/bin/bash
wget http://IP_DE_VOTRE_SE3/interfaces
mv /etc/network/interfaces /etc/network/interfaces.old
mv interfaces /etc/network/
/etc/init.d networking restart
```

- déposer dans le répertoire **/home/netlogon/clients-linux/unefois/^nomade-/** du serveur ledit script
- lancer l'intégration au domaine de vos clients

Les ordinateurs de la classe nomade redémarreront une première fois après l'intégration au domaine : laissez les branchés en filaire. Lors de leur premier boot en version « intégrés », les clients récupéreront le fichier de configuration du réseau et se connecteront automatiquement au réseau wifi adéquat.

Le jour où vous aurez besoin de faire d'importantes mises à jour, vous pourrez tout aussi facilement refaire momentanément basculer ces postes en filaire...

12 Un mot sur les imprimantes

Ne disposant personnellement d'aucune imprimante réseau, je n'ai jamais pu tester ce qui suit [23]. Je suis donc loin de maîtriser l'aspect « gestion des imprimantes » sur les clients GNU/Linux. Ceci étant, il faut bien évoquer ce point très important.

Sur un client GNU/Linux, le répertoire **/mnt/netlogon/divers/** contient un sous-répertoire nommé **imprimantes/** qui vous permettra de stocker de manière centralisée des fichiers **.ppd** (pour « PostScript Printer Description ») qui sont des sortes de drivers permettant d'installer des imprimantes sur les clients GNU/Linux. Vous pouvez télécharger de tels fichiers (qui dépendent du modèle de l'imprimante) sur ce site par exemple :

http://www.openprinting.org/printers

Supposons que, dans le répertoire **/mnt/netlogon/divers/imprimantes/**, se trouve le fichier **.ppd** d'un modèle d'imprimante réseau donné, vous pouvez alors lancer son installation sur un client GNU/Linux via la commande suivante (en tant que **root**) :

```
lpadmin -p NOM-IMPRIMANTE -v socket://IP-IMPRIMANTE:9100 \<Touche ENTRÉE>
    -E -P /mnt/netlogon/divers/imprimantes/fichier.ppd
```

Cette commande doit être en principe exécutée une seule fois sur le client GNU/Linux. Si tout va bien, vous devriez ensuite [24] être en mesure d'imprimer tout ce que vous souhaitez à travers vos applications

22. Changez bien sûr **IP_DE_VOTRE_SE3**... par l'IP réelle de votre Se3...
23. Si vous avez du code bash à me proposer pour automatiser l'installation des imprimantes sur les clients GNU/Linux via par exemple la fonction **initialisation_perso**, je suis preneur (francois.lafont@crdp.ac-versailles.fr).
24. Même après redémarrage du système.

favorites (navigateur Web, traitement de texte, lecteur de PDF etc). Si plusieurs imprimantes sont installées sur un client, pour faire en sorte que l'imprimante **NOM-IMPRIMANTE** soit l'imprimante par défaut, il faut exécuter en tant que **root** :

```
lpadmin -d NOM-IMPRIMANTE
```

Et pour supprimer l'imprimante :

```
lpadmin -x NOM-IMPRIMANTE
```

13 Désinstallation/réinstallation du paquet se3-clients-linux

13.1 Désinstallation complète

Si jamais vous souhaitez désinstaller complètement le paquet **se3-clients-linux** de votre serveur, rien de plus simple. En tant que **root** sur le serveur, il suffit de lancer la commande suivante :

```
apt-get purge se3-clients-linux
```

Et c'est tout. Une fois la commande ci-dessus exécutée, votre serveur ne garde **plus la moindre trace** d'installation du paquet **se3-clients-linux**.

Attention, en désinstallant le paquet de la sorte (avec **apt-get purge**), tout le répertoire **/home/netlogon/clients-linux/** du serveur (et tout ce qu'il contient) sera effacé. Si vous aviez pris la peine de vous concocter un fichier **logon_perso** à votre sauce, d'écrire de nombreux scripts dans le répertoire **unefois/** etc., tout sera purement et simplement effacé.

13.2 Désinstallation partielle en vue d'une réinstallation

Avec la commande ci-dessous (où l'instruction **purge** est remplacée par **remove**), les choses se passent un peu différemment :

```
apt-get remove se3-clients-linux
```

Le paquet **se3-clients-linux** est bien désinstallé comme dans le cas précédent, sauf que le répertoire **/home/netlogon/clients-linux/** du serveur n'est pas totalement effacé. Tous les fichiers/répertoires que vous avez le droit de modifier seront conservés, si bien que l'arborescence du répertoire ressemblera à ceci :

```
-- clients-linux/
   |-- bin/
   |   `-- logon_perso
   |-- distribs/
   |   |-- precise/
   |   |   `-- skel/
   |   `-- squeeze/
   |       `-- skel/
   |-- divers/
   `-- unefois/
```

Ainsi, après réinstallation du paquet, vous retrouverez inchangés :

- le fichier **logon_perso** ;
- tous les profils distants de chaque distribution prise en charge ;
- le contenu du répertoire **divers/** ;
- le contenu du répertoire **unefois/**.

En résumé, une réinstallation du paquet **se3-clients-linux** avec conservation des fichiers/dossiers modifiables se fait ainsi :

```
apt-get remove se3-clients-linux
apt-get install se3-clients-linux
```

Une telle réinstallation du paquet peut être utile si jamais, pour une raison ou pour une autre, vous avez commis un certain nombre de modifications malheureuses en voulant « hacker » certains fichiers du paquet et que vous souhaitez repartir de zéro sans pour autant perdre vos fichiers « personnels [25] ». Autre cas où la réinstallation peut être utile : lors d'une mise à jour du paquet (auquel cas d'ailleurs il sera plus naturel d'exécuter la commande « **apt-get dist-upgrade** »). Dans ce cas aussi, les fichiers « personnels » seront conservés en l'état.

> **Remarque :** ici, la notion de fichiers/répertoires « modifiables » ou « personnels » n'est pas à prendre au pied de la lettre. Dans l'absolu, vous pouvez tenter de modifier ce que vous voulez dans les fichiers du paquet **se3-clients-linux**. Simplement, sont considérés comme « modifiables » (ou « personnels ») seulement les fichiers/répertoires **conservés** lors d'une réinstallation ou d'une mise à jour du paquet.

14 Signaler un problème, faire une remarque etc.

Étant donné que le paquet **se3-clients-linux** fait partie du projet SambaÉdu, le mieux à faire pour signaler un problème, faire une remarque etc. est de passer par la liste de diffusion SambaÉdu samba-edu@mailinglistes.tice.ac-caen.fr [26]. N'hésitez pas à me signaler toute erreur. Si vous envoyez un message sur cette liste avec un objet assez évocateur (par exemple avec l'expression « clients GNU/Linux » dedans), il y a peu de chance que je passe à côté. J'essayerai alors de vous répondre et, dans la mesure du possible, de rectifier le problème.

15 Contribuer à améliorer le paquet

Dans votre coin, vous avez réussi à modifier le paquet (un script d'intégration, le script de logon etc.) afin d'en étendre les fonctionnalités (par exemple afin de prendre en charge une nouvelle distribution, un autre gestionnaire de bureau qui a votre faveur et qui n'est pas actuellement pris en charge par le paquet etc. etc. etc.) ? Si c'est le cas, n'hésitez surtout pas à nous en faire part sur la liste de diffusion se3-devel@listes.tice.ac-caen.fr. Nous serons ravis d'intégrer au paquet vos contributions pour peu que vous les ayez déjà testées avant [27].

16 Les évolutions du paquet

Ci-dessous, vous trouverez le contenu du fichier **changelog** du paquet. C'est une sorte de journal qui décrit les modifications du paquet au fur et à mesure du temps :

25. Ce qu'on appelle les fichiers « personnels », ce sont les fichiers que vous avez le droit de modifier, ceux qui sont en vert dans l'arborescence située à la section 3 page 9.
26. Attention, pour pouvoir écrire à cette liste de diffusion, il faut d'abord s'y inscrire : http://listes.tice.ac-caen.fr/mailman/listinfo/samba-edu.
27. Car tester une fonctionnalité prend du temps (la coder aussi d'ailleurs), et c'est le temps qui nous limite dans l'élaboration du paquet et non un manque de volonté bien sûr.

Le 20 août 2012 version 1.0
C'est le début.

Le 29 août 2012 version 1.0.1
Modification du script d'intégration pour la distribution Precise afin qu'il fonctionne aussi avec Xubuntu "12.04". Petite modification de la documentation en conséquence.

Le 30 août 2012 version 1.0.2
Modification dans le profil par défaut (aussi bien pour Squeeze que pour Precise) proposé par le paquet. Le lanceur "Google" supposait que Firefox était installé sur les clients. Ce lanceur a été remplacé par un lanceur de type "Link" et c'est le navigateur par défaut qui se lance automatiquement (qui peut être Firefox ou non).

Le 3 novembre 2012 version 1.0.3
Modification du fichier postinst du paquet : suppression du « exit 1 » en cas d'erreur sur le test du serveur NTP du SE3. Ce « exit 1 » est trop rude et peut coincer le gestionnaire de paquets.

Ajout de la dépendance « se3-logonpy > 2.3.7216 » pour résoudre le conflit avec ce paquet au niveau du fichier smb-CIFS.conf.

Le 6 novembre 2012 version 1.1
Modification de la structure du paquet via la suppression du fichier LISEZMOI.TXT du paquet .deb, sachant que ce fichier est maintenant créé lors de l'appel du script postinst.
Modification des fichiers d'intégration afin qu'on ne puisse pas intégrer une machine dont le nom fait plus de 15 caractères.
Mais le plus gros changement est le transfert de toute la partie montage des partages du fichier logon vers le fichier logon_perso à l'aide de fonctions « user-friendly ». Ainsi, l'administrateur pourra gérer la partie montage des partages (suivant le compte, suivant la station cliente, suivant l'appartenance à un groupe etc.) comme il le souhaite et le tout de manière relativement simple.

Le 24 novembre 2012 version 1.1.1
Ajout de la fonction changer_papier_peint utilisable dans le fichier logon_perso et qui fait ce qu'on attend d'elle.
Ajout du fichier connexion_ssh_serveur.bash qui est un simple raccourci pour lancer une connexion ssh au sur le serveur à partir d'un client Linux.

Modifications des lanceurs Google.desktop situés dans les profils par défaut de Squeeze et de Precise afin d'ajouter le shebang. Dans le fichier logon, au moment de l'ouverture de session, ajout d'un « chmod u+x » sur tous les fichiers de la forme "$REP_HOME/Bureau/"*.desktop au cas où...

Ajout des options « port=139,netbiosname=$NOM_HOTE » dans le montage des partages avec "mount.cifs". La conséquence est que la variable de substitution %m utilisée dans les fichiers smb*.conf est bien remplacée par le nom (netbios) du client Linux. Cela impliquera peut-être une possibilité d'utiliser les imprimantes du serveur Samba. Peut-être...

Le 6 janvier 2013 version 1.1.2

Ajout du fichier reconfigure.bash afin que l'on puisse reconfigurer les bons droits sur les fichiers du paquet, et que l'on puisse injecter le contenu de logon_perso dans logon, le tout directement via un client Linux avec le compte admin. Modification de la documentation en conséquence.

Le 6 janvier 2013 version 1.1.3

Modification du fichier preinst du paquet car deux vérifications un peu zélées interrompaient directement un « apt-get upgrade » si celui-ci correspondait simultanément à une mise à jour du paquet se3-clients-linux et à une mise à jour de certains paquets Se3 (précisément les paquets se3 et se3-domain).

Le 9 décembre 2013 version 1.1.6

Ajout de l'option de montage "sec=ntlmv2" car suite à une MAJ sur Precise, sans cette option les montages ne se font plus.

Ajout de l'option "--bwlimit=2048" dans le rsync entre le skel par défaut du serveur et le skel local. L'idée est de limiter le débit du rsync afin, *on l'espère*, de garantir une meilleure stabilité et un meilleur taux de réussite de la synchronisation.

Désormais, si la synchronisation des skel ne se fait pas, alors le .VERSION local est remis dans son état initial afin que la synchrosation soit retentée la prochaine fois sans qu'on ait besoin d'éditer le fichier .VERSION du serveur.

Prise en charge de Debian Wheezy au niveau de l'intégration. La désintégration ne fonctionne pas pour l'instant et le script a été mis dans /home/netlogon/clients-linux/.test.
