DOCUMENTATION TECHNIQUE

Script de correction automatique

des lanceurs Wine multi-utilisateurs

Objectif du document

Ce document présente le fonctionnement et le déploiement d'un script Bash conçu pour corriger automatiquement les lanceurs d'applications Wine sur un poste Linux multi-utilisateurs. Il détaille le problème rencontré, la logique du script, l'explication de chaque composant technique, ainsi qu'un exemple concret du résultat obtenu.

1. Contexte et problématique

Wine est une couche de compatibilité qui permet d'exécuter des applications Windows sur Linux. Lorsqu'une application Wine est installée, elle génère automatiquement un fichier de lanceur (.desktop) contenant le chemin absolu vers le dossier .wine de l'utilisateur qui a effectué l'installation.

Ce mécanisme fonctionne bien pour un poste mono-utilisateur, mais devient problématique dans un environnement scolaire multi-utilisateurs : le lanceur créé par un utilisateur référence son propre dossier personnel, ce qui le rend inutilisable pour tous les autres comptes.

⚠ Problème	Le lanceur .desktop généré par Wine contient un chemin en dur (ex : /home/jim/.wine/...). Si un autre utilisateur (ex : matheo) tente de lancer l'application, Wine cherche le dossier /home/jim/.wine qui n'existe pas dans sa session → l'application ne se lance pas.

2. Objectif du script

Le script Bash développé ici résout ce problème de manière entièrement automatique. Il s'exécute à l'ouverture de session et adapte tous les lanceurs Wine au nom de l'utilisateur connecté, sans aucune intervention manuelle.

- Parcourt tous les fichiers .desktop de l'utilisateur
- Cible uniquement ceux qui contiennent une référence à Wine
- Remplace /home/ancien_utilisateur/.wine par /home/utilisateur_connecté/.wine
- Ne modifie que le chemin utilisateur, le reste de la ligne Exec= est préservé
- Fonctionne pour n'importe quelle application Wine, quel que soit l'utilisateur

✓ Résultat	Chaque utilisateur dispose de lanceurs Wine fonctionnels et adaptés à son compte dès l'ouverture de session, sans aucune configuration manuelle.

3. Script Bash – Code complet annoté

Voici le script complet avec des commentaires détaillés sur chaque bloc de code.

```
#!/bin/bash
# ========================================
# Correction automatique des lanceurs Wine
# Adaptation du chemin utilisateur dans Exec=
# ========================================

# Dossier contenant les lanceurs .desktop de l'utilisateur
DESKTOP_DIR="$HOME/.local/share/applications"

# Récupère le nom de l'utilisateur actuellement connecté
CURRENT_USER="$USER"
```

```
# Vérification : le dossier des lanceurs existe-t-il ?
if [ ! -d "$DESKTOP_DIR" ]; then
    echo "Dossier des lanceurs introuvable"
    exit 1
fi

# Boucle sur tous les fichiers .desktop du dossier
for file in "$DESKTOP_DIR"/*.desktop; do

    # On vérifie que le fichier contient Wine (insensible à la casse)
    if grep -qi "wine" "$file"; then

        # Remplacement du chemin utilisateur dans la ligne Exec=
        sed -i "s|/home/[^/]*\.wine|/home/$CURRENT_USER/.wine|g" "$file"
        echo "Lanceur corrigé : $(basename "$file")"
    fi

done

echo "Correction des lanceurs Wine terminée"
```

4. Explication détaillée bloc par bloc

4.1 – Déclaration des variables

Variable / Élément	Valeur exemple	Description
$DESKTOP_DIR	*~/.local/share/applications*	Chemin vers le dossier contenant les lanceurs .desktop de l'utilisateur
$CURRENT_USER	*matheo*	Nom de l'utilisateur actuellement connecté, récupéré via $USER
$HOME	*/home/matheo*	Dossier personnel de l'utilisateur connecté (variable système)
$USER	*matheo*	Nom de l'utilisateur connecté (variable système automatique)

i Pourquoi $USER ?	La variable $USER est définie automatiquement par le système à l'ouverture de session. Elle contient toujours le nom de l'utilisateur connecté, ce qui rend le script universel sans aucune modification manuelle.

4.2 – Vérification du dossier des lanceurs

Avant de commencer le traitement, le script vérifie que le dossier des lanceurs existe bien. Si ce n'est pas le cas, il affiche un message d'erreur et s'arrête proprement (exit 1) pour éviter des erreurs en cascade.

```
if [ ! -d "$DESKTOP_DIR" ]; then
    echo "Dossier des lanceurs introuvable"
    exit 1
fi
```

Élément	Signification
[! -d ...]	Test : le dossier n'existe PAS (-d = directory, ! = négation)
exit 1	Arrêt du script avec un code d'erreur (1 = erreur, 0 = succès)

4.3 – Boucle sur les fichiers .desktop

La boucle for parcourt tous les fichiers ayant l'extension .desktop présents dans le dossier des lanceurs. Pour chaque fichier, la commande grep vérifie si le mot « wine » apparaît (l'option -qi rend la recherche insensible à la casse).

```
for file in "$DESKTOP_DIR"/*.desktop; do
  if grep -qi "wine" "$file"; then
    ...
  fi
done
```

Option / Commande	Description
grep -q	Recherche silencieuse (pas d'affichage) — retourne vrai/faux
grep -i	Recherche insensible à la casse (Wine = wine = WINE)
*.desktop	Sélectionne tous les fichiers .desktop du dossier
$(basename ...)	Extrait uniquement le nom de fichier, sans le chemin complet

4.4 – La commande sed : cœur du script

La commande sed est responsable du remplacement du chemin utilisateur dans la ligne Exec= de chaque lanceur Wine.

```
sed -i "s|/home/[^/]*\.wine|/home/$CURRENT_USER/.wine|g" "$file"
```

Élément sed	Description
-i	Modifie le fichier directement (in-place), sans créer de copie
s|...|...|g	Syntaxe de substitution avec | comme délimiteur (évite les conflits avec /)
/home/[^/]*	/home/ + n'importe quel nom d'utilisateur (regex : [^/]* = tout sauf /)
\.wine	Le dossier .wine (le \ échappe le point qui est un caractère spécial en regex)
/home/$CURRENT_USER/.wine	Chemin de remplacement avec le nom de l'utilisateur connecté
g	Remplacement global (toutes les occurrences sur la ligne, pas seulement la 1ère)

ℹ Pourquoi | et non / ?	La commande sed utilise le caractère | comme séparateur au lieu du / habituel, car les chemins de fichiers contiennent déjà des /. Utiliser / comme séparateur provoquerait des erreurs de syntaxe.

5. Exemple concret – Avant / Après

Voici un exemple réel avec l'application BiblioManuels, installée par l'utilisateur 'jim' et utilisée ensuite par 'matheo'.

✗ AVANT	✓ APRÈS
`Exec=wine "/home/jim/.wine/drive_c/Program Files (x86)/Sejer/BiblioManuels/BiblioManuels.exe"` *⚠ Le chemin contient 'jim' en dur → le lanceur est cassé pour tout autre utilisateur.*	`Exec=wine "/home/matheo/.wine/drive_c/Program Files (x86)/Sejer/BiblioManuels/BiblioManuels.exe"` *✓ Le chemin est adapté à 'matheo' automatiquement → lanceur fonctionnel.*

6. Déploiement du script

Pour que le script s'exécute automatiquement à l'ouverture de chaque session utilisateur, il doit être intégré au mécanisme de démarrage de session (session_startup.sh).

7. Synthèse et bonnes pratiques

✓	Ne jamais coder un nom d'utilisateur en dur dans un script — toujours utiliser $USER ou $HOME
✓	Utiliser grep -qi pour une détection robuste et insensible à la casse
✓	Préférer le délimiteur | dans sed lorsque les chaînes à traiter contiennent des /
✓	Toujours vérifier l'existence du dossier cible avant d'effectuer des opérations dessus
✓	Tester le script manuellement sur un seul fichier avant un déploiement global