Retour de Projet Multics

CHALOYARD Lucas

April 20, 2021

1 Introduction

Pour ce projet d'étude Multics j'ai eu l'occasion de prendre en main le système Multics au travers du simulateur fourni.
Je ferai donc un point sur l'utilisation de Multics, son administration et les outils fournis et utiles pour l'implémentation des démonstrateurs.

2 Administration

Pour ce qui est de l'administration du système Multics, j'ai principalement utilisé les consignes retrouvables sur ce lien : https://multics-wiki.swenson.org/index.php/Administering_Multics

J'ai donc eu l'occassion de tester la création de compte, ainsi que la création de projet qui fonctionnenent bien en utilisant les consignes données sur le lien précédent.

L'ajout d'utilisateur a un projet fonctionne aussi correctement.

Dans le cadre du démonstrateur, nous supposerons l'utilisation d'un utilisateur nommé **DemoUser** (important pour les scripts).
On peut aussi créer un projet dédié pour la Demo (ce qui est plus sécurisé que de faire tout cela dans SysAdmin), on pourrait imaginer que ce projet se nomme **Demo**.

3 Demonstrateur

3.1 Git et création

Tout les fichiers devant être porté sur le système Multics pour le démonstrateur se trouveront sur ce repo Github : https://github.com/LuK74/MulticsDemoACL.

IMPORTANT, lors ce que le démonstrateur sera mis en place, les fichiers, les répertoires et donc toute l'architecture du dossier Demo doit être construite par un l'User utilisé pour la démo.

IMPORTANT, pour les besoins de la démo, les scripts partent du principe que le nom de l'User utilisé pour la démo est **DemoUser**.
Si vous souhaitez en utiliser un autre, certaines modifications doivent être apporté à certains script du dossier **ACL**

3.2 Script de présentation

J'ai implémenté plusieurs scripts de présentation des features des ACL.
Ces scripts auront pour objectif d'afficher du texte permettant d'expliquer la feature présenté à l'utilisateur. Quelques manipulations seront aussi effectuées par le script afin de présenter de manière plus concrète à l'utilisateur la feature présenté.

3.3 Programme acl_indicator du Playground

Ce programme intéractif permet à un utilisateur de tester simplement l'ouverture de divers fichiers, il aura simplement à passer le chemin d'acès vers le fichier auquel il veut accèder, et le mode demandé, et une ouverture sera effectuée.

Cela permet de tester facilement différents mode d'ouverture (read ou write), et cela proposera aussi à l'utilisateur une entrée ACL simple pour faire fonctionner l'ouverture du fichier si cela ne fonctionne pas.

Cela permet aussi de tester simplement et rapidement les ACL des fichiers sur lesquels on effectuait ces tests.

3.4 Fonctionnement

Pour l'instant les scripts ne sont pas reliés un à un, il est nécessaire pour l'utilisateur de les exécuter lui-même (utilisation de la commande **ec** ou **exec_com**).
Afin de s'assurer que tout outil sera compilé, et ajouter au search path, il est nécessaire d'éxecuter le script **mainACL.ec** au démarrage.

Afin d'assurer un bon fonctionnement, il sera aussi nécessaire que la hiérarchie suivantes soient bien respectée.

3.5 Hiérarchie

```
Demo> (répertoire principal)
| -> mainACL.ec
| -> Tools>
| -> ACL>

Demo>Tools> (outils utilisé pour les scripts)
| -> compileWaitEntry.ec
| -> wait_entry (crée par script)
| -> waitEntry.c

Demo>ACL> (contient les script faisant des courtes demos et explications)
| -> Directories.ec
| -> ComplexeACL.ec
| -> Demo.ec
| -> MulticsACLFeatures3.ec
| -> MulticsACLFeatures2.ec
| -> MulticsACLFeatures.ec
| -> UnixToMultics.ec
| -> Playground/

Demo>ACL>Playground> (répertoire pour effectuer différent test sur des fichiers lambda)
| -> acl_indicator (crée par script)
| -> IndicatorCompile.ec
| -> indicatorACL.c
| -> setupPlayground.ec (génére quelques fichiers pour le playground)
| -> * (fichiers sur lesquels l'user peut effectuer des tests)
```

3.6 ACL Répertoire

Voici une liste des features présentées par chacun des fichiers du répertoire ACL.

- Directories.ec : Spécificité des modes sur les Directories (différent d'Unix, sur Multics on a les modes "s", "m", et "a"
- ComplexeACL.ec : Démonstration de création de contrainte complexe possible simplement grâce au ACL
- Demo.ec : Simple démonstration du bon fonctionnement des ajouts d'ACL

- UnixToMultics.ec : Démonstration montrant qu'il est simple de convertir les permissions Unix en permissions Multics
- MulticsACLFeatures.ec : Démonstration de l'utilisation des "*", afin de créer des entrées ACL générales
- MulticsACLFeatures2.ec : Démonstration de comment on peut restreindre les droits d'utilisateurs spécifiques même si une autre entrée est supposé leur donner plus de droit
- MulticsACLFeatures3.ec : Démonstration de l'ordonnancement des entrées ACL et de pourquoi cet ordre là

ATTENTION, les 3 scripts MulticsACLFeatures devraient être exécuté à la suite et dans l'ordre numéroté (pas de numéro = 1) afin d'avoir une demo fonctionnelle

3.7 À améliorer/rajouter

- Liaison entre les différents scripts **ec** afin d'avoir une démonstration ordonnée et automatisée
- Amélioration du **Playground** grâce à l'utilisation d'un langage autre que C (car ses fonctionnalitées sont très limitées sur cette version du simulateur, lib stdlib et unistd non présentes)
- Ajouter des démonstration sur les Ring de protection
- Chercher une manière pour switcher d'utilisateur dans une même session de simulateur afin de pouvoir créer des script testant les ACL et/ou Ring du point de vue de 2 utilisateurs différents
- OU créer des scripts au-dessus du simulateur afin de créer plusieurs sessions Multics et alterner entre elle afin de faire des démonstrations plus explicites (plus compliqué)
- **Comment automatiser la création des fichiers/répertoire de démo**, car pour l'instant il faut retaper à la main chacun d'entre eux.
  Cela peut-être très simple mais je n'ai pas encore regardé

4 Problèmes rencontrées

Voici une liste des problèmes rencontrées afin que les prochaines personnes travaillant sur ce projet puisse gagner du temps

- Langage C très limité sur Multics. Pas de stdlib ou unistd donc impossible de faire de la création de processus, exécution de programme, etc...
- Étudier plus le langage de scripting (exec_com), peu détaillé sur le wiki mais assez détaillé dans le système Multics (>doc>info>ec.info, ou utilisation de help).
- Se pencher sur la création automatique d'user grâce au fichier ".ini" du simulateur. Celui créer par le script "site_setup.sh" dans le répertoire QuickStart ne fonctionnait pas de mon côté

5 Aide

5.1 Liste de liens utiles

- Administration du système Multics : https://multics-wiki.swenson.org/index.php/Administering_Multics#Creating_a_New_Project
- Utilisation du compilateur C : https://multics-wiki.swenson.org/index.php/Multics_C_Compiler
- Utilisation du système Multics : https://multics-wiki.swenson.org/index.php/Using_Multics
- Commandes équivalent Multics des commandes Unix : https://multics-wiki.swenson.org/index.php/Linux-to-Multics_Command_Mapping

- Lien du wiki sur le simulateur utilisé pour Multics (même lien qu'au-dessus) : https://multics-wiki.swenson.org/index.php/Main_Page
- Explication de la sécurité sur Multics (ACL et Ring) : https://multicians.org/multics-data-security.html

5.2 Liste commandes Multics utiles

Aussi trouvable ici, et bien d'autre : https://multics-wiki.swenson.org/index.php/Linux-to-Multics_Command_Mapping)

- cwd : Change working directory, équivalent de cd sous Unix.
  "cwd <" permet de remonter d'un étage dans la hiérarchie.
  "cwd >" ramène à la racine.
  "cwd" ramène à l'équivalent d'un home directory.
- ls : Liste les segments (fichiers), rajouter le "-a" pour voir les Directories.
- la (list_acl) : Liste les ACL sur le fichier/répertoire donné en path
- da (delete_acl) : Supprimer les ACL sur le fichier/répertoire donné en path
- sa (set_acl) : Permet d'ajouter des entrées ACL sur le fichier/répertoire donné en path
- create : Créer un fichier
- create_dir : Créer un répertoire
- delete : Supprime un fichier
- delete_dir : Supprime un répertoire
- asr (add_search_rules) : Ajoute un répertoire dans le recherche path lorsqu'on demande l'exécution d'une commande (équivalent d'ajouter une entrée dans la variable d'environnement PATH sous Unix).
- cc : Compilateur C
- lint : Linter C
- ec (exec_com) : Exécute les scripts .ec

5.3 Liste raccourci Emacs utiles

- CTRL+X puis CTRL+S : Sauvegarde
- CTRL+X puis CTRL+C : Quitter Emacs
- CTRL+N : Passe à la ligne suivante
- CTRL+P : Passe à la ligne précédente
- CTRL+F : Avance d'un caractère
- CTRL+B : Recule d'un caractère
- CTRL+Q : Permet d'insérer certain caractère spéciaux demandant l'usage de AltGr (ex : #)