Projet Minix

Professeur : xxx
Assistants : Christophe Paasch et Fabien Duchêne

LINGI1113 : Système Informatique 2

Université Catholique de Louvain

Martin Donies
Florentin Rochet

2011-2012

Table des matières

0.1 Introduction

Nous avons réalisé correctement les appels systèmes correspondant aux ressources suivantes :

RLIMIT NICE
RLIMIT NPROC
RLIMIT NOFILE
RLIMIT FSIZE

Pour faciliter nos explications et votre compréhension de notre Minix modifié, nous allons lister et expliquer dans ce rapport les éléments du patch généré par la commande make dist.

0.2 Architecture globale

0.3 Implémentation des limites

0.3.1 RLIMIT_NICE

0.3.2 RLIMIT_NPROC

0.3.3 RLIMIT_FSIZE

Pour être certain que la limite à la taille des fichiers ne soit jamais dépassée, nous vérifions à chaque appel de write que la somme de la taille du fichier actuel. Pour la fonction truncate, nous observons simplement la taille à la quelle le processus souhaite tronquer le fichier.

Pour gérer l'erreur, nous envoyons un signal SIGXFSZ que nous avons ajouté dans include/signal.h. De cette façon, si le processus gère le signal, il pourra continuer à s'exécuter et l'appel système retournera une erreur EFBIG. Si il ne le gère pas, le processus sera arrêté de force.

0.3.4 RLIMIT_NOFILE

Pour gérer cette limite, nous avons implémenté un compteur qui compte le nombre de file descriptor liés au processus. Il aurait été plus simple et moins risqué de vérifier le nombre de fichiers ouverts dans la fonction get_fd(). Mais nous serions passés d'une complexité temporelle en O(1) à O(n) (où n représente la taille du tableau contenant les file descriptors).

En début de projet, nous envisagions d'enlevé les limites définies par les constantes de minix pour les remplacer par un système de. ainsi les tableaux auraient été adaptés aux limites parametrées par setrlimit().