Projet Minix

Professeur : Marc Lobel
Assistants : Christophe Paasch et Fabien Duchêne

LINGI1113 : Système Informatique 2

Université Catholique de Louvain

Martin Donies
Florentin Rochet

2011-2012

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.

2 Architecture globale

3 listing et explications des modifications

Voici pour commencer l'implémentation des appels systèmes demandés.

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/lib/libc/other/
       _getrlimit.c ./src/lib/libc/other/_getrlimit.c
2  --- ./src_orig/lib/libc/other/_getrlimit.c      1970-01-01
       01:00:00.000000000 +0100
3  +++ ./src/lib/libc/other/_getrlimit.c   2012-05-11 20:09:56.000000000
       +0200
```

Code correspondant à la fonction getrlimit(int resource, struct rlimit* rlim).

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/lib/libc/other/
       _setrlimit.c ./src/lib/libc/other/_setrlimit.c
2  --- ./src_orig/lib/libc/other/_setrlimit.c      1970-01-01
       01:00:00.000000000 +0100
3  +++ ./src/lib/libc/other/_setrlimit.c   2012-05-11 23:52:15.000000000
       +0200
```

Code correspondant à la fonction setrlimit(int resource, struct rlimit* rlim)
Ces appels systèmes permettent de contacter le serveur pm et le serveur vfs. On va donc lister les différences dans ces deux serveurs en commençant par le server pm.

```
1  Binary files ./src_orig/servers/pm/a.out and ./src/servers/pm/a.out
       differ\\
2  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/pm/forkexit.c
       ./src/servers/pm/forkexit.c\\
3  --- ./src_orig/servers/pm/forkexit.c     2010-07-02 14:41:19.000000000
       +0200\\
4  +++ ./src/servers/pm/forkexit.c 2012-05-13 11:36:49.000000000 +0200\\
```

```
1  -  if ((procs_in_use == NR_PROCS) ||
2  -              (procs_in_use >= NR_PROCS-LAST_FEW && rmp->mp_effuid !=
       0))
3  +  if ((procs_in_use == NR_PROCS) || (is_full_limit()) ||
```

```
4       +                       (procs_in_use >= NR_PROCS-LAST_FEW && rmp->mp_effuid !=
            0))
```

Ce code permet de vérifier si la limite du nombre de process est atteinte. Elle est vérifiée durant un fork. la méthode is_full_limit() est implémentée dans pm/rlimit.c