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

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().

0.4 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  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/pm/forkexit.c
       ./src/servers/pm/forkexit.c
2  --- ./src_orig/servers/pm/forkexit.c    2010-07-02 14:41:19.000000000
       +0200
3  +++ ./src/servers/pm/forkexit.c 2012-05-13 11:36:49.000000000 +0200
4
5  -   if ((procs_in_use == NR_PROCS) ||
6  -               (procs_in_use >= NR_PROCS-LAST_FEW && rmp->mp_effuid !=
       0))
7  +   if ((procs_in_use == NR_PROCS) || (is_full_limit()) ||
8  +               (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

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/pm/main.c ./
       src/servers/pm/main.c
2  --- ./src_orig/servers/pm/main.c        2010-07-13 17:30:17.000000000
       +0200
3  +++ ./src/servers/pm/main.c     2012-05-13 19:08:44.000000000 +0200
4
5  +       /*
6  +       * Execute do_getrlimit ou do_setrlimit suivant le cas. result
       contient OK ou une erreur.
7  +       * */
8  +       case GET_RES_LIMIT : result = do_getrlimit(&m_in,mp); setreply(
       who_p,result); break;
9  +       case SET_RES_LIMIT : result = do_setrlimit(&m_in,mp); setreply(
       who_p,result); break;
10 +
11 .
12 .(etc)
13 .
14   /* Initialize process table, including timers. */
15    for (rmp=&mproc[0]; rmp<&mproc[NR_PROCS]; rmp++) {
16       init_timer(&rmp->mp_timer);
17 +       rmp->nice_cur_ceiling = 0;
18 +       rmp->nice_ceiling = PRIO_MAX-1;
19 +       rmp->nproc_cur_ceiling = 200;
20 +       rmp->nproc_ceiling = NR_PROCS-1;
21 }
```

Les cas GET_RES_LIMIT et SET_RES_LIMIT du switch permettent de catcher l'appel système envoyé par _getrlimit.c et _setrlimit.c. Cela n'est pas obligatoire étant donné que le fichier table.c permet de faire un appel automatique si call_nr arrive dans le default du switch. Nous l'avons fait ici pour pouvoir utiliser setreply et renvoyer le résultat à la fonction appelante (0 ou une erreur)

La suite du code (avec la boucle for) permet l'initialisation des variables que nous avons rajouté dans la structure mproc. nice_cur_ceiling et nproc_cur_ceiling représentent la rlim_cur pour leur syscall respectif, nice_ceiling et nproc_ceiling la rlimit_max. Le choix de l'initialisation est arbitraire, tout en restant logique.

```
1       diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/pm/misc.c ./
        src/servers/pm/misc.c
2       --- ./src_orig/servers/pm/misc.c          2010-07-02 14:41:19.000000000
        +0200
3       +++ ./src/servers/pm/misc.c      2012-05-13 19:08:44.000000000 +0200
4       @@ -446,6 +446,9 @@
5            * the kernel's scheduling queues.
6            */
7
8  +         /* handle de nice value limit*/
9  +         if(rlimit_nice(arg_pri,arg_who) == EINVAL)
10 +                 return(EINVAL);
11           if ((r = sched_nice(rmp, arg_pri)) != OK) {
12                   return r;
13           }
```

rlimit_nice est une fonction déclarée dans rlimit.c, elle permet de vérifier que la nice value respecte les bornes lorsqu'elle est modifiée. Si non, on retourne EINVAL

```
1 +++ ./src/servers/pm/rlimit.c    2012-05-13 19:08:44.000000000 +0200
```

Fichier important où l'on a implémenté do_getrlimit, do_setrlimit et différentes autres fonction pour le bon fonctionnement de nos modifications dans le process manager. Un fichier rlimit.c est également présent dans le serveur vfs et remplis un rôle identique.

Passons maintenant aux modifications dans le serveur vfs.

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/vfs/filedes.c ./
       src/servers/vfs/filedes.c
2  --- ./src_orig/servers/vfs/filedes.c     2010-08-30 15:44:07.000000000 +0200
3  +++ ./src/servers/vfs/filedes.c 2012-05-13 19:08:43.000000000 +0200
4  @@ -139,6 +139,7 @@
5           if(fproc[f].fp_pid == PID_FREE) continue;
6           for(fd = 0; fd < OPEN_MAX; fd++) {
7                   if(fproc[f].fp_filp[fd] && fproc[f].fp_filp[fd] == fp) {
8  +                        rm_open_file(&fproc[f]);
9                           fproc[f].fp_filp[fd] = NULL;
10                          n++;
11                  }
12 @@ -221,6 +222,7 @@
13
14                  /* Found a free slot, add descriptor */
15                  FD_SET(j, &fproc[proc].fp_filp_inuse);
16 +                if(!add_open_file(&fproc[proc])) return(EMFILE); /* check
       limit of nbr of file and increment nbr of files */
17                  fproc[proc].fp_filp[j] = cfilp;
18                  fproc[proc].fp_filp[j]->filp_count++;
```

```
19                      return j;
20  @@ -281,6 +283,9 @@
21            /* Found a valid descriptor, remove it */
22            FD_CLR(fd, &fproc[proc].fp_filp_inuse);
23            fproc[proc].fp_filp[fd]->filp_count--;
24  +
25  +         rm_open_file(&fproc[proc]);
26  +
27            fproc[proc].fp_filp[fd] = NULL;
28
29            return fd;
```

rm_open_file(struct froc *) est une fonction déclaré dans vfs/rlimit.c qui permet de décrementer le compteur utilisé pour le nombre de file descriptor. add_open_file(struct froc*) quant à elle incrémente le même compteur lorsque qu'un nouveau file descriptor est créé. Elle permet aussi de vérifier que la limite (déclarée dans fproc) n'est pas dépassée. Ces fonctions sont utilisées ailleurs avec la même logique. Les autres localisations sont données dans le patch et ne seront pas détaillé dans ce rapport.

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/vfs/link.c ./src/
       servers/vfs/link.c
2  --- ./src_orig/servers/vfs/link.c        2010-08-30 15:44:07.000000000 +0200
3  +++ ./src/servers/vfs/link.c    2012-05-13 19:08:43.000000000 +0200
4  @@ -193,7 +193,7 @@
5     */
6      struct vnode *vp;
7      int r;
8  -
9  +   if (is_too_big(m_in.flength,0)) return(EFBIG);
10     if ((off_t) m_in.flength < 0) return(EINVAL);
```

La function is_too_big permet de vérifier la limite imposée pour la taille des fichiers (limite déclarer dans fproc) Cette fonction est utilisée a d'autres avec la même logique. Ceux-ci peuvent être retrouvé dans le patch et ne seront donc pas détaillé dans ce rapport.

```
1  diff -urN --exclude .svn --exclude '*~' ./src_orig/servers/vfs/main.c ./src/
       servers/vfs/main.c
2  --- ./src_orig/servers/vfs/main.c        2010-07-22 16:55:28.000000000 +0200
3  +++ ./src/servers/vfs/main.c    2012-05-13 19:08:43.000000000 +0200
4  @@ -249,6 +249,10 @@
5            rfp->fp_grant = GRANT_INVALID;
6            rfp->fp_blocked_on = FP_BLOCKED_ON_NONE;
7            rfp->fp_revived = NOT_REVIVING;
8  +         rfp->fopen_cur_ceiling = 100;
9  +         rfp->fopen_hard_ceiling = OPEN_MAX;
10 +         rfp->fsize_cur_ceiling = 20971520;
11 +         rfp->fsize_hard_ceiling = 52428800;
```

Initialisation des différentes limites dans la structures fproc. Ces limites sont données arbitrairement, tout en faisant attention qu'elles ne provoquent pas d'erreur systèmes (plafond trop bas pour fsize par exemple)

```
1  +++ ./src/servers/vfs/rlimit.c  2012-05-13 19:08:43.000000000 +0200
```

Fichier important, tout comme dans le serveur pm, il contient l'implémentation de do_getrlimit et do_setrlimit ainsi que d'autres fonction nécessaire au bon fonctionnement de l'imposition des limites.