GL0-2001 Systèmes d'exploitation **H16**

RAPPORT – Travail Pratique no 1
Utilisation de Linux, programmation thread

fait par: Gaël Dostie 111096568

[X] IFT [ ] GLO [ ] GIF [ ] Autre

Alexandre Picard-Lemieux 111103625

[X] IFT [ ] GLO [ ] GIF [ ] Autre

Résultat: _____ sur 100

1. Répertoires et fichiers

1.1 Examiner un répertoire

```
etu1@ift2001-desktop:~$ cd /
etu1@ift2001-desktop:/$ ls -l
   …
drwxr-xr-x  17 root root  3680 2016-01-30 13:57 dev
   …
drwxr-xr-x   6 root root  4096 2011-08-22 14:25 home
   …
drwxrwxrwt  11 root root  4096 2016-01-30 13:58 tmp
drwxr-xr-x  11 root root  4096 2016-01-30 13:41 usr
   …
```

Les colonnes du résultat représentent les choses suivantes (dans l'ordre de gauche à droite) : type de fichier et les modes associés, le nombre de hard links, le nom du propriétaire, le nom du groupe d'utilisateur auquel il appartient, la taille, la date de modification, l'heure de la modification, le nom du fichier.

1.2 Protection

```
etu1@ift2001-desktop:/home$ mkdir nouvelleEntree
mkdir: impossible de créer le répertoire «nouvelleEntree»:
Permission non accordée
```

Si on regarde l'entrée home de la question 1.1, on voit que le répertoire home appartient à root et qu'il faut posséder les privilèges de root pour y faire des modifications.

```
etu1@ift2001-desktop:/tmp$ mkdir nouvelleEntree
etu1@ift2001-desktop:/tmp$ ls
keyring-3k65Yy  nouvelleEntree  orbit-etu1  orbit-gdm  pulse-
PKdhtXMmr18n  ssh-OHMJmE1367  vmware-etu1
```

Si on regarde l'entrée tmp de la question 1.1, on voit que le répertoire tmp appartient à root, mais peux être écrit par n'importe qui, d'où les 'w'.

1.3 Modification des droits d'accès

```
etu1@ift2001-desktop:/home$ chmod a-rwx,u=rwx etu1
etu1@ift2001-desktop:/home$ chmod a-rwx,u=rwx /usr/local
chmod: modification des permissions de «/usr/local»: Opération
non permise
```

Cela fonctionne sur etu1, car le propriétaire de etu1 est etu1. Dans le cas de /usr/local, c'est root qui est le propriétaire, donc on ne peut pas modifier ses droits d'accès.

1.4 Répertoires "bin"

```
etu1@ift2001-desktop:/$ ls /bin
bash     bzexe         cat    dash                 dir
egrep       gunzip    less      login   mount       nc.openbsd
ntfs-3g.usermap  ps        run-parts   static-sh  touch
vdir          zfgrep
bunzip2  bzfgrep       chgrp  date                 dmesg
false       gzexe     lessecho  ls      mountpoint  netcat
open            pwd
   …
etu1@ift2001-desktop:/$ ls /usr/bin
[                           defoma-psfont-installer
getent                     ld
pdb2.6                                sensible-editor
vmware-toolbox
2to3                        defoma-subst
gethostip                  ld.bfd
pdf2dsc
…
```

ls dans /bin
mkdir dans /bin
cat dans /bin

g++ dans /usr/bin
gedit dans /usr/bin

```
etu1@ift2001-desktop:/$ find /bin -name ls -print
/bin/ls
```

ls

1.5 Répertoire "dev"

etu1@ift2001-desktop:~$ find /dev -name "sda*" -print

```
/dev/sda5
/dev/sda2
/dev/sda1
/dev/sda
```

Cette commande permet de trouver les 4 périphériques matériels communs qui sont présent dans ce répertoire qui correspond est le disque dur qui est divisé en 3 partitions.

1.6 Répertoire "include"

Le fichier « stdio.h » a été trouvé.

```
etu1@ift2001-desktop:~$ ls -l /usr/include/ | grep stdio.h
-rw-r--r--  1 root root  31109 2011-01-21 18:51 stdio.h
```

Le fichier est le en-tête standard d'entrée et sortie écrit en C.

1.7 Répertoire "proc"

model name	: Intel(R) Core(TM) i7-3667U CPU @ 2.00GHz
cpu MHz		: 3200.000

Il n'y a qu'un processeur dans la machine virtuelle.

```
u1@ift2001-desktop:~$ ps
  PID TTY          TIME CMD
 3121 pts/0    00:00:00 bash
 3139 pts/0    00:00:00 ps

etu1@ift2001-desktop:~$ cd /proc/3121
etu1@ift2001-desktop:/proc/3121$ ls
attr            cpuset   io        mountinfo   pagemap      smaps    task
auxv            cwd      latency   mounts      personality  stack
wchan
```

```
cgroup          environ  limits    mountstats  root        stat
clear_refs      exe      loginuid  net         sched       statm
cmdline         fd       maps      oom_adj     schedstat   status
coredump_filter fdinfo   mem       oom_score   sessionid   syscall

etu1@ift2001-desktop:/proc/1701$ cat status
Name: bash
State:   S (sleeping)

etu1@ift2001-desktop:/proc/3121$ cat status
Name: bash
State:   S (sleeping)
voluntary_ctxt_switches:   118

etu1@ift2001-desktop:/proc/3121$ cat status
Name: bash
State:   S (sleeping)
voluntary_ctxt_switches:   121

etu1@ift2001-desktop:/proc/3121$ cat status
Name: bash
State:   S (sleeping)
voluntary_ctxt_switches:   125
```

1.8 Fichier "math.h"

```
etu1@ift2001-desktop:/$ find / -name math.h
/usr/include/math.h
etu1@ift2001-desktop:/$ gedit /usr/include/math.h
```

1.57079632679489661923

1.9 Fichiers cachés

```
etu1@ift2001-desktop:/$ cd ~
etu1@ift2001-desktop:~$ ls –al
total 180
drwx------ 26 etu1 etu1 4096 2016-01-30 20:13 .
drwxr-xr-x  6 root root 4096 2011-08-22 14:25 ..
   …
drwxr-xr-x 10 etu1 etu1 4096 2010-09-25 16:30 .config
   …
drwx------  6 etu1 etu1 4096 2010-09-25 16:32 .gnome2
```

```
   …
drwx------  4 etu1 etu1 4096 2010-09-25 15:55 .mozilla
   …
```

2. Compilation et exécution

2.1 Création d'un fichier source

```
#include <unistd.h>
#include <stdio.h>

int main() {
    printf(" Bonjour! Le numero de l'utilisateur est %d\n",
getuid());
    printf(" numero du processus est %d\n", getpid());
    printf(" - nom Alexandre Picard-Lemieux\n");
    printf(" - nom Gael Dostie\n");
    return 0;
}
```

2.2 Compilation

```
etu1@ift2001-desktop:~$ gcc -o qui-suis-je qui-suis-je.c
```

2.3 Exécution d'un programme

```
etu1@ift2001-desktop:~$ ./qui-suis-je
 Bonjour! Le numero de l'utilisateur est 1003
 numero du processus est 3473
 - nom Alexandre Picard-Lemieux
- nom Gael Dostie

etu1@ift2001-desktop:~$ ps -o ppid=3473
 3473
 3119
```

```
 3208

etu1@ift2001-desktop:~$ ps
  PID TTY          TIME CMD
 3208 pts/1    00:00:00 bash
 3479 pts/1    00:00:00 ps
```

Donc le numéro du processus parent est le 3208 qui est le bash.

2.4 Appels systèmes fait par un programme : `strace`

```
etu1@ift2001-desktop:~$ strace ./qui-suis-je
execve("./qui-suis-je", ["./qui-suis-je"], [/* 39 vars */]) = 0
brk(0)                                  = 0x8c49000
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
mmap2(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0xb787f000
access("/etc/ld.so.preload", R_OK)      = -1 ENOENT (No such file or
directory)
open("/etc/ld.so.cache", O_RDONLY)      = 3
fstat64(3, {st_mode=S_IFREG|0644, st_size=54737, ...}) = 0
mmap2(NULL, 54737, PROT_READ, MAP_PRIVATE, 3, 0) = 0xb7871000
close(3)                                = 0
access("/etc/ld.so.nohwcap", F_OK)      = -1 ENOENT (No such file or
directory)
open("/lib/tls/i686/cmov/libc.so.6", O_RDONLY) = 3
read(3,
"\177ELF\1\1\1\0\0\0\0\0\0\0\0\0\3\0\3\0\1\0\0\0000m\1\0004\0\0\0"...,
512) = 512
fstat64(3, {st_mode=S_IFREG|0755, st_size=1405508, ...}) = 0
mmap2(NULL, 1415592, PROT_READ|PROT_EXEC, MAP_PRIVATE|MAP_DENYWRITE, 3, 0)
= 0x793000
mprotect(0x8e6000, 4096, PROT_NONE)     = 0
mmap2(0x8e7000, 12288, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_DENYWRITE, 3, 0x153) = 0x8e7000
mmap2(0x8ea000, 10664, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_FIXED|
MAP_ANONYMOUS, -1, 0) = 0x8ea000
close(3)                                = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0xb7870000
set_thread_area({entry_number:-1 -> 6, base_addr:0xb78706c0,
limit:1048575, seg_32bit:1, contents:0, read_exec_only:0,
```

```
limit_in_pages:1, seg_not_present:0, useable:1}) = 0
mprotect(0x8e7000, 8192, PROT_READ)     = 0
mprotect(0x8049000, 4096, PROT_READ)    = 0
mprotect(0x30d000, 4096, PROT_READ)     = 0
munmap(0xb7871000, 54737)               = 0
getuid32()                              = 1003
fstat64(1, {st_mode=S_IFCHR|0620, st_rdev=makedev(136, 1), ...}) = 0
mmap2(NULL, 4096, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0)
= 0xb787e000
write(1, " Bonjour! Le numero de l'utilisa"..., 46 Bonjour! Le numero de
l'utilisateur est 1003
) = 46
getpid()                                = 3504
write(1, " numero du processus est 3504\n", 30 numero du processus est
3504
) = 30
write(1, " - nom Alexandre Picard-Lemieux\n", 32 - nom Alexandre Picard-
Lemieux
) = 32
write(1, " - nom Gael Dostie\n", 19 - nom Gael Dostie
)    = 19
exit_group(0)                           = ?
```

Le premier appel système est : execve("./qui-suis-je", ["./qui-suis-je"], [/* 39 vars */]) = 0
Cette appel sert à exécuter le programme qui-suis-je.

Le deuxième appel système est : brk(0) = 0x8c49000
Cette appel sert à changer la taille de donnée de la segmentation

Il y a 14 appels systèmes distincts qui sont fait par le programme.

Par déduction, l'appel système invoqué par printf est write.

2.5 Utilisation pratique de `strace`

```
etu1@ift2001-desktop:~/Documents/remise$ strace ./ProgrammeMystere
open("Configuration.xml", O_RDONLY)     = -1 ENOENT (No such file or
directory)
```

Il n'est pas capable de trouver le fichier Configuration.xml.

2.6 Inspection d'un *core dump*

```
int main() {
    int * pointeur = 0;
     int value = *pointeur;
    return 0;
}

etu1@ift2001-desktop:~$ gcc -g plante.c -o plante

etu1@ift2001-desktop:~$ ./plante
Erreur de segmentation (core dumped)

etu1@ift2001-desktop:~$ gdb plante core
Program terminated with signal 11, Segmentation fault.
#0  0x080483c4 in main () at plante.c:3
3       int value = *pointeur;
```

3. Fonction fork

```
#include <unistd.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>

int main(int argc, char* argv[]) {
    int pid;
    pid = fork();

     if(pid==0) {
        printf("Bonjour! Je suis le fils %d\n",getpid());
          sleep(120);
        exit(0);
    } else {
          printf("Coucou! Je suis le parent, et mon fils a le numéro de
processus %d\n",pid);
        wait(0);
    }
}
```

```
gcc question3.c -o question3
./question3
Coucou! Je suis le parent, et mon fils a le numéro de processus 5337
Bonjour! Je suis le fils 5337

ps -a
  PID TTY          TIME CMD
 5336 pts/0    00:00:00 question3
 5337 pts/0    00:00:00 question3
 5377 pts/12   00:00:00 ps
```

4. Exercice de programmation par threads

Listing du fichier ChercherSHA256.c :

```
#include <string.h>
#include "sha256.h"
#include <stdlib.h>
#include <stdio.h>
#include <time.h>
#include <sys/stat.h>
#include <pthread.h>

#define NB_THREADS 4

int flag = 0;

typedef struct {
    int startOffset;
    int endOffset;
    unsigned char * target;
}Params;

/* Pour calculer un intervalle de temps */
struct timespec diff(struct timespec start, struct timespec end);


void * search(void * args ) {
   Params *params = (Params *) args;
   FILE *pFichier;
   sha256_context ctx;
   char *pLigne;
   size_t longueur = 0;
```

```
   ssize_t read;
   unsigned char sha256sum[32];
   long positionFin;

   /* Ouverture du dictionnaire */
   pFichier = fopen("mots.txt","r");

   /* On va chercher la ligne de fin donné par endOffset */
   if (fseek(pFichier, params->endOffset, SEEK_SET) != 0) {
      printf("Problème lors du déplacement dans le fichier.\n");
   }
   positionFin = ftell(pFichier);

   if (fseek(pFichier, params->startOffset, SEEK_SET) != 0) {
      printf("Problème lors du déplacement dans le fichier.\n");
   }

   if (pFichier == NULL) {
      printf("Le dictionnaire mots.txt est manquant! \n");
      exit(EXIT_FAILURE);
   }

   pLigne = (char *)malloc(longueur);

   while ((read = getline(&pLigne, &longueur, pFichier)) != -1 && flag != 1
&& ftell(pFichier) <= positionFin) {
      /* On enleve le retour de chariot '\n' a la fin */
      if (pLigne[read-1] == '\n') pLigne[read-1] = '\0';

      /* On initialise la fonction de hashing */
       sha256_starts( &ctx );

       /* On lui passe maintenant caractere par caractere le string a signer
*/
       sha256_update( &ctx, (uint8 *)pLigne, strlen(pLigne));

      /* On termine le tout */
       sha256_finish( &ctx, sha256sum );

       /* On compare le string maintenant */
       int Match = 1;
       int j = 0;
       for (j = 0; j < 32; j++ ) {
         if (sha256sum[j]!=params->target[j]) {
```

```
            Match = 0;
            break;
          }
        }

        if (Match) {
          printf("Bingo! Le mot est %s.\n",pLigne);
          flag = 1;
          break;
        }
    }

    free(pLigne);
    fclose(pFichier);
    pthread_exit(NULL);
}

int main( int argc, char *argv[] )
{

     int i, j;
     char output[65];
     unsigned char sha256sumtarget[32];
     struct stat st;
     ssize_t read;
     Params params[NB_THREADS];
     pthread_t threads[NB_THREADS];


     read = strlen(argv[1]);
     if (read!=64) {
       /* La signature en hexadecimal doit etre 64 caracteres */
          printf("La signature %s n'a que %d caracteres\n!",argv[1],read);
       exit(EXIT_FAILURE);
     }

     printf("La signature recherchee est %s\n",argv[1]);

    char tmp[3];
    tmp[2] = '\0'; /* Null-terminated string */
     /* Conversion du string en un entier de 32 octets */
     for (i=0;i<32; i++) {
       tmp[0] = argv[1][2*i];
```

```
      tmp[1] = argv[1][2*i+1];
        sscanf(tmp,"%hhx",&sha256sumtarget[i]);
    }

    struct timespec time1, time2;
     int temp;
     clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time1);

    stat("mots.txt",&st);
    int size = st.st_size;
    int sizeSplitByThreads = size / NB_THREADS;
    int cpt = 0;
    for (i=0; i < NB_THREADS; i++) {
        params[i].startOffset = cpt;
        params[i].endOffset = cpt + sizeSplitByThreads - 1;
        params[i].target = sha256sumtarget;
        cpt += sizeSplitByThreads;

        if (i == NB_THREADS -1 ) {
            params[i].endOffset = size;
        }

        int status  = pthread_create(&threads[i], NULL,search,(void *)&
params[i]);

        if( status != 0) {
            printf("Erreur, code : %d\n",status);
            exit(EXIT_FAILURE);
        }
    }

    for(i=0; i < NB_THREADS; i++) {
        pthread_join(threads[i], NULL);
    }

   clock_gettime(CLOCK_PROCESS_CPUTIME_ID, &time2);
   struct timespec deltaTime = diff(time1,time2);
   printf("Temps de calcul : %ld.%ld
seconde\n",deltaTime.tv_sec,deltaTime.tv_nsec);


    exit(EXIT_SUCCESS);
}
```

```
struct timespec diff(struct timespec start, struct timespec end)
{
   struct timespec temp;
   if ((end.tv_nsec-start.tv_nsec)<0) {
      temp.tv_sec = end.tv_sec-start.tv_sec-1;
      temp.tv_nsec = 1000000000+end.tv_nsec-start.tv_nsec;
   } else {
      temp.tv_sec = end.tv_sec-start.tv_sec;
      temp.tv_nsec = end.tv_nsec-start.tv_nsec;
   }
   return temp;
}
```

Sortie d'écran lors de l'exécution de test.sh

```
etu1@ift2001-desktop:~/Documents/tp1/q4$ sh test.sh
La signature recherchee est
41b7d74cf4c20fa187720966c0a80d14bd54d5837b0210e699a3e1d76406c3e9
Bingo! Le mot est ATTENTION.
Temps de calcul : 0.74180750 seconde
La signature recherchee est
e5571c8af72b91933c52c111b891b05c85ce4424e24b934ef46c71eea3bdb36a
Bingo! Le mot est CLAVIER.
Temps de calcul : 0.203975701 seconde
La signature recherchee est
4db9ca9ad94283998256449a67b9e5074d1069ee3e3405d75d21e865fd68b767
Bingo! Le mot est SUBTERFUGE.
Temps de calcul : 0.162428261 seconde
La signature recherchee est
99c4a24ed90bec0a6b3b76142ec777fcea0a35332c12d4364c326beebbbe82be
Bingo! Le mot est WAGON.
Temps de calcul : 0.242669651 seconde
etu1@ift2001-desktop:~/Documents/tp1/q4$
```