Noyau d'un système d'exploitation INF2610

Annexe 1

Département de génie informatique et génie logiciel

POLYTECHNIQUE MONTRÉAL
AFFILIÉE À L'UNIVERSITÉ DE MONTRÉAL

Hiver 2014

Exemple 5.1 : Processus, threads noyau, threads utilisateur

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/wait.h>
static unsigned long long a = 0;
#define MAX 1000000000
void *count(void *arg) ;
int main( ) {
        int p, i;
        for (i = 0; i < 2; i++) {
        if ((p = fork()) < 0) return 1;
        if (p == 0) {   count(&a);
          printf("child %d a=%llu\n", getpid(), a);
          return 0;
         }
        }
        for (i = 0; i < 2; i++) { wait(NULL);}
        printf("a=%llu \n", a);
        return 0;
}
```

```
void *count(void *arg) {
  volatile unsigned long long*var = (unsigned long long*) arg;
  volatile unsigned long long i;
  for (i = 0; i < MAX; i++)
        *var = *var + 1;
  return NULL;
}
```

```
jupiter$ g++ thread_process.cpp –o thread_process
jupiter$ time ./thread_process
child 10629 a=1000000000
child 10630 a=1000000000
a=0

real  0m3.187s
user 0m6.215s
sys   0m0.005s
jupiter$
```

Exemple 5.2 : Processus, threads noyau, threads utilisateur

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <pthread.h>
static unsigned long long a = 0;
#define MAX 1000000000
void *count(void *arg) ;

int main( ) {
        int p, i;
        pthread_t t1, t2;
        pthread_create(&t1, NULL, count, &a);
        pthread_create(&t2, NULL, count, &a);
        pthread_join(t1, NULL);
        pthread_join(t2, NULL);
        printf("pid=%d a=%llu\n", getpid(), a);
        return 0;
}
```

```
void *count(void *arg) {
  volatile unsigned long long*var = (unsigned long long*) arg;
  volatile unsigned long long i;
  for (i = 0; i < MAX; i++)
        *var = *var + 1;
  return NULL;
}
```

```
jupiter$ g++ thread_pthread.cpp –o thread_pthread
jupiter$ time ./thread_pthread
pid=10686 a=987341287

real  0m10.258s
user 0m20.338s
sys   0m0.010s
jupiter$
```

```
jupiter$ time ./thread_pthread
pid=10695 a=1019958670

real  0m10.740s
user 0m21.423s
sys   0m0.013s
jupiter$
```

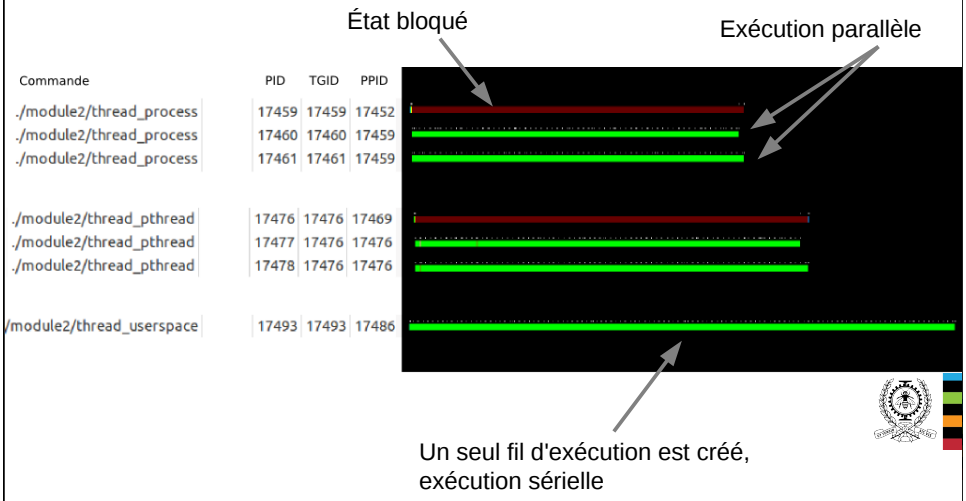

Exemple 5.3 : Processus, threads noyau, threads utilisateur

```
#include <unistd.h>
#include <stdlib.h>
#include <stdio.h>
#include <sys/types.h>
#include <pth.h>
static unsigned long long a = 0ULL;
#define MAX 1000000000
void *count(void *arg);

int main(int argc, char **argv) {
        pth_init();
        pth_t t1, t2;
        t1 = pth_spawn(PTH_ATTR_DEFAULT, count, &a);
        t2 = pth_spawn(PTH_ATTR_DEFAULT, count, &a);
       pth_join(t1, NULL);
        pth_join(t2, NULL);
        printf("a=%llu\n", a);
        return EXIT_SUCCESS;
}
```

```
void *count(void *arg) {
  volatile unsigned long long*var = (unsigned long long*) arg;
  volatile unsigned long long i;
  for (i = 0; i < MAX; i++)
        *var = *var + 1;
  return NULL;
}
```

```
jupiter$ g++ thread_userspace.cpp –o thread_userspace –lpth
jupiter$ time ./thread_userspace
a=2000000000

real  0m3.608s
user 0m3.599s
sys   0m0.001s
jupiter$
```

Exemple 5.4 : Processus, threads noyau, threads utilisateur



Exemple 6.1 : printf avant fork

```
#include <unistd.h>          /* pour write */
#include <stdio.h>           /* pour printf */
int main( ) {

    printf(" ici 1er printf de %d ", getpid());
    write(1," ici 1er write  ",16);
    printf(" ici 2eme printf de  %d ", getpid());
    fork();
    write(1," ici 2eme write  ", 17);
    printf("end of line printf de %d\n", getpid());
    write(1, " ici 3eme write \n",17);
    return 0;

}
```

```
jupiter$./printfwrite
 ici 1er write   ici 2eme write   ici 1er printf de 11243  ici 2eme printf de  11243 end of line printf de 11243
 ici 3eme write
 ici 2eme write   ici 1er printf de 11243  ici 2eme printf de  11243 end of line printf de 11244
 ici 3eme write
jupiter$
```

Exemple 6.2 : printf avant fork

```
#include <unistd.h>            /* pour write */
#include <stdio.h>             /* pour printf  et fflush*/
int main( ) {

    printf(" ici 1er printf de %d ", getpid());
    write(1," ici 1er write  ",16);
    printf(" 2eme printf de  %d ", getpid());
    fflush(stdout);
    fork();
    write(1," ici 2eme write  ", 17);
    printf("end of line printf de %d\n", getpid());
    write(1, " ici 3eme write \n",17);
    return 0;

}
```

```
jupiter$./printfwrite
 ici 1er write   ici 1er printf de 11258  ici 2eme printf de  11258  ici 2eme write  end of line printf de 11258
 ici 3eme write
 ici 2eme write  end of line printf de 11259
 ici 3eme write
jupiter$
```