test/makethreads.c

```
/*
 * Programme C pour tester le threads, avec un nombre important de threads,
 * des threads qui en lancent d'autres, des pointeurs donnes en argument, etc.
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */

#include "syscall.h"

char* stringAvecEntier( char* str, int entier, int position )
{
        if( entier < 100 )
        {
                if( entier > 9 )
                {
                        str[ position + 0 ] = entier / 10 + '0';
                }
                else
                {
                        str[ position + 0 ] = ' ';
                }

                str[ position + 1 ] = entier % 10 + '0';
        }

        return str;
}

void f( void* a )
{
        int ptr = *((int*)a);
        PutString( stringAvecEntier( "f est appele avec:    \n", ptr, 20 ) );

//      UserThreadExit();
}

void g( void* arg )
{
        int a = 5;
        int val=(int)arg;

        PutString(stringAvecEntier("g est appele avec:    \n",val,20));

        --a;
        UserThreadCreate(f,&a);

        if(val>0)
            UserThreadCreate( g, (void*)(val-1) );

//      UserThreadExit();
}

void ping( void* a )
{
        int i;
        int c;

        for( i = 0 ; i < 10 ; i++ )
        {
                c = i * 3;
                PutString( stringAvecEntier( "ping:    \n",  c, 6 ) );
        }

//      UserThreadExit();
}

void pong( void* a )
{
        int i;

        for( i = 0 ; i < 10 ; i++ )
        {
                PutString( stringAvecEntier( "pong:    \n",  i, 6 ) );
        }

//      UserThreadExit();
}

int main ()
{
        int a = 10;

        PutString( "demarrage1...\n" );

        UserThreadCreate( g, (void*)(a-5) );

        PutString( "demarrage2...\n" );

        UserThreadCreate( f, (void*)(&a) );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

<< ... etc ... >>

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        PutString( "\nfin\n" );

//      Halt();

        return 0;
}
```

test/malloc.c

```
/*
 * Programme C pour tester malloc et free
 *
 * Par Savas Ali TOKMEN
 */

#include "syscall.h"

void ecrire( int* mem )
{
        mem[ 0 ] = 4;
        mem[ 1 ] = 45;
        mem[ 2 ] = 456;
}

int main ()
{
        int* mem;

        PutString( "Allocation...\n" );

        mem = (int*) AllocEmptyPage();

        PutString( "Ecriture dans ");
        PutInt( ( int ) mem );
        PutString( " (donc la page " );
        PutInt( ( ( int ) mem ) / 128 );
        PutString( ")\n");

        ecrire( mem );

        PutString( "Lecture:\n" );
        PutString( "  Quatre: " );
        PutInt( mem[ 0 ] );
        PutString( "\n  Quarante cinq: " );
        PutInt( mem[ 1 ] );
        PutString( "\n  Quatre cent cinquante six: " );
        PutInt( mem[ 2 ] );
        PutString( "\n\n" );

        mem = (int*) AllocEmptyPage();

        PutString( "Ecriture dans ");
        PutInt( ( int ) mem );
        PutString( " (donc la page " );
        PutInt( ( ( int ) mem ) / 128 );
        PutString( ")\n");

        ecrire( mem );

        PutString( "Lecture:\n" );
        PutString( "  Quatre: " );
        PutInt( mem[ 0 ] );
        PutString( "\n  Quarante cinq: " );
        PutInt( mem[ 1 ] );
        PutString( "\n  Quatre cent cinquante six: " );
        PutInt( mem[ 2 ] );
        PutString( "\n\n" );

        PutString( "Free de la page ");
        PutInt( ( ( int ) mem ) / 128 );
        PutString( "\n");

        FreePage( mem );

        PutString( "Essai d'ecriture a la page " );
        PutInt( ( ( int ) mem ) / 128 );
        PutString( " (devrait planter)\n" );

        ecrire( mem );

        PutString( "Fin\n" );

        return 0;
}
```

test/mem_protect.c

```
/*
 * Tests pour la reecriture memoire a des adresses invalides
 *
 * Par Savas Ali TOKMEN
 */

#include "syscall.h"

int main ()
{
        int i;
        int* start = 0;

        PutString( "je me suicides !!\n" );

        for( i = 0 ; i < 2048 ; i++ )
        {
                start[ i ] = 0;
        }

        PutString( "suicide OK\n" );

        return 0;
}
```

test/semaphores.c

```
/*
 * Programme C pour tester les semaphores
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */

#include "syscall.h"

sem_t sema;

void ping( void* a )
{
        int i;

        for( i = 0 ; i < 10 ; i++ )
        {
                Sema_P( sema );

                PutString( "ping: " );
                PutInt( i );
                PutString( "\n" );

                Sema_V( sema );
        }
}

void pong( void* a )
{
        int i;

        for( i = 0 ; i < 10 ; i++ )
        {
                Sema_P( sema );

                PutString( "pong: " );
                PutInt( i );
                PutString( "\n" );

                Sema_V( sema );
        }
}

int main ()
{
        PutString( "Id de la semaphore: " );
        PutInt( sema = Sema_Init( 1 ) );
        PutString( "\n\n" );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        return 0;
}
```

test/semaphores_local.c

```
/*
 * Programme C pour tester les semaphores n'utilisant que les variables locaux
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */

#include "syscall.h"

void ping( void* a )
{
        int i;
        int sema = (int) a;

        for( i = 0 ; i < 10 ; i++ )
        {
                Sema_P( sema );

                PutString( "ping: " );
                PutInt( i );
                PutString( "\n" );

                Sema_V( sema );
        }
}
void pong( void* a )
{
        int i;
        int sema = (int) a;

        for( i = 0 ; i < 10 ; i++ )
        {
                Sema_P( sema );

                PutString( "pong: " );
                PutInt( i );
                PutString( "\n" );

                Sema_V( sema );
        }
}
int main ()
{
        sem_t sema;

        PutString( "Id de la semaphore: " );
        PutInt( sema = Sema_Init( 1 ) );
        PutString( "\n\n" );

        UserThreadCreate( ping, (void*) sema );
        UserThreadCreate( pong, (void*) sema );

        UserThreadCreate( ping, (void*) sema );
        UserThreadCreate( pong, (void*) sema );

        UserThreadCreate( ping, (void*) sema );
        UserThreadCreate( pong, (void*) sema );

        return 0;
}
```

test/userpages.c

```
/*
 * Programme C pour tester le multi-processus
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */

#include "syscall.h"

int main ()
{
        PutString("userpages demarre...\n");

        ForkExec( "test/userpages0" );
        ForkExec( "test/userpages1" );

        ForkExec( "test/userpages0" );
        ForkExec( "test/userpages1" );

        ForkExec( "test/makethreads" );
        ForkExec( "test/userpages1" );
        ForkExec( "test/userpages1" );

        /* oui, ceci va recurser jusqu'a saturation de memoire !! */
        ForkExec( "test/userpages" );

        ForkExec( "test/makethreads" );
        ForkExec( "test/userpages0" );
        ForkExec( "test/userpages1" );
        ForkExec( "test/userpages1" );
        ForkExec( "test/userpages1" );


        PutString("userpages a fini!\n");

        return 0;
}
```

test/userpages0.c

```
/*
 * Programme C pour tester le multi-processus avec quelques threads
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */
#include "syscall.h"

char* stringAvecEntier( char* str, int entier, int position )
{
        if( entier < 100 )
        {
                if( entier > 9 )
                {
                        str[ position + 0 ] = entier / 10 + '0';
                }
                else
                {
                        str[ position + 0 ] = ' ';
                }

                str[ position + 1 ] = entier % 10 + '0';
        }

        return str;
}
void ping( void* a )
{
        int i;
        int c;

        for( i = 0 ; i < 10 ; i++ )
        {
                c = i * 3;
                PutString( stringAvecEntier( "ping:    \n",  c, 6 ) );
        }
}
void pong( void* a )
{
        int i;

        for( i = 0 ; i < 10 ; i++ )
        {
                PutString( stringAvecEntier( "pong:    \n",  i, 6 ) );
        }
}
int main ()
{
        PutString( "demarrage...\n" );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        UserThreadCreate( ping, 0 );
        UserThreadCreate( pong, 0 );

        PutString( "\nfin\n" );

        return 0;
}
```

test/userpages1.c

```
/*
 * Deuxieme programme C pour tester le multi-processus avec quelques threads
 *
 * Par Nicolas MOBILIA, Remi BARRAQUAND, Romain FORNARA et Savas Ali TOKMEN
 */

#include "syscall.h"

char* stringAvecEntier( char* str, int entier, int position )
{
        if( entier < 100 )
        {
                if( entier > 9 )
                {
                        str[ position + 0 ] = entier / 10 + '0';
                }
                else
                {
                        str[ position + 0 ] = ' ';
                }

                str[ position + 1 ] = entier % 10 + '0';
        }

        return str;
}
void pang( void* a )
{
        int i;
        int c;

        for( i = 0 ; i < 10 ; i++ )
        {
                c = i * 3;
                PutString( stringAvecEntier( "pang:    \n",  c, 6 ) );
        }
}
void peng( void* a )
{
        int i;

        for( i = 0 ; i < 10 ; i++ )
        {
                PutString( stringAvecEntier( "peng:    \n",  i, 6 ) );
        }
}
int main ()
{
        PutString( "DEMARRAGE 1...\n" );

        UserThreadCreate( pang, 0 );
        UserThreadCreate( peng, 0 );

        UserThreadCreate( pang, 0 );
        UserThreadCreate( peng, 0 );

        PutString( "\nFIN 1\n" );

        return 0;
}
```

Sortie partielle de userpages:

```
$ userprog/nachos -rs -x test/userpages

userpages demarre...
demarrage...
DEMARRAGE 1...
demarrage...

fin
demarrage1...
demarrage2...
DEMARRAGE 1...
userpages demarre...
demarrage1...
demarrage...
DEMARRAGE 1...

FIN 1
userpages a fini!

<< ... etc ... >>

pong:  5
f est appele avec:   4
peng:  8
ping: 27
pong:  9
pong:  6
peng:  9
pong:  7
pong:  8
pong:  9
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages0" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages0" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/makethreads" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages" !
Pas assez de memoire pour charger le fichier "test/makethreads" !
Pas assez de memoire pour charger le fichier "test/userpages0" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Pas assez de memoire pour charger le fichier "test/userpages1" !
Machine halting!

Ticks: total 1031621, idle 770368, system 203710, user 57543
Disk I/O: reads 0, writes 0
Console I/O: reads 0, writes 8000
Paging: faults 0
Network I/O: packets received 0, sent 0

Cleaning up...
```