Lab03a - Threads

Nome:Hao Yue Zheng-10408948
Nome:Smuel Zheng-10395781
Nome:Vitor Pasquarelli Cinalli-10401806

Linker de `Git`
https://github.com/mauriciohao/Sistema-operacional.git

EXERCÍCIO: Incremente o exemplo acima para representar a troca de informações de contexto entre a thread e o processo pai.

Codigo

```
#define _GNU_SOURCE
#include <stdlib.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sched.h>
#include <stdio.h>

// Define o tamanho da pilha como 64kB
#define FIBER_STACK 1024*64

// Variável compartilhada entre o processo pai e a thread filha
volatile int shared_variable = 0;

// Função que será executada pela thread filha
int threadFunction(void* argument)
{
   printf("Thread filha está rodando e modificando shared_variable\n");
   shared_variable = 42;  // Modifica a variável compartilhada
   printf("Thread filha terminando\n");
   return 0;
}

int main()
{
   void* stack;
   pid_t pid;

   // Aloca a pilha
   stack = malloc(FIBER_STACK);
   if (stack == 0)
   {
      perror("malloc: não foi possível alocar a pilha");
      exit(1);
   }

   printf("Criando thread filha\n");
```

```
    // Chama a syscall clone para criar a thread filha
    pid = clone(&threadFunction, (char*) stack + FIBER_STACK,
        SIGCHLD | CLONE_FS | CLONE_FILES | CLONE_SIGHAND | CLONE_VM, 0);
    if (pid == -1)
    {
        perror("clone");
        exit(2);
    }

    // Aguarda o término da thread filha
    pid = waitpid(pid, 0, 0);
    if (pid == -1)
    {
        perror("waitpid");
        exit(3);
    }

    // Verifica o valor de shared_variable após a modificação pela thread filha
    printf("Thread filha retornou e shared_variable agora é %d\n", shared_variable);

    // Libera a pilha
    free(stack);
    printf("Pilha liberada.\n");

    return 0;
}
```

Solução

```
  GNU nano 5.8
#define _GNU_SOURCE
#include <stdlib.h>
#include <malloc.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <signal.h>
#include <sched.h>
#include <stdio.h>

// Define o tamanho da pilha como 64kB
#define FIBER_STACK 1024*64

// Variável compartilhada entre o processo pai e a thread filha
volatile int shared_variable = 0;

// Função que será executada pela thread filha
int threadFunction(void* argument)
{
    printf("Thread filha está rodando e modificando shared_variable\n");
    shared_variable = 42;  // Modifica a variável compartilhada
    printf("Thread filha terminando\n");
    return 0;
}

int main()
{
    void* stack;
    pid_t pid;

    // Aloca a pilha
    stack = malloc(FIBER_STACK);
    pid_t pid;

    // Aloca a pilha
    stack = malloc(FIBER_STACK);
    if (stack == 0)
    {
^G Help          ^O Write Out     ^W Where Is      ^K Cut           ^T Execute       ^
```

```
[cloudshell-user@ip-10-132-38-160 ~]$ gcc -o run trecho.c
[cloudshell-user@ip-10-132-38-160 ~]$ ./run
Creating child thread
child thread exiting
Child thread returned and stack freed.
[cloudshell-user@ip-10-132-38-160 ~]$ sudo nano
[cloudshell-user@ip-10-132-38-160 ~]$ gcc -o  run1
.bash_logout   .bashrc        .config/       lab2          lab3a.c        mack.pem.txt   pipes.c        proj1-o.c      run1           trecho.c
.bash_profile  chaveweb.pem   exemplo2.c     lab2c.c       .local/        .pem.txt       proj1.c        run            .ssh/          .zshrc
[cloudshell-user@ip-10-132-38-160 ~]$ gcc -o  run1 lab3a.c
[cloudshell-user@ip-10-132-38-160 ~]$ ./run1
Criando thread filha
Thread filha está rodando e modificando shared_variable
Thread filha terminando
Thread filha retornou e shared_variable agora é 42
Pilha liberada.
[cloudshell-user@ip-10-132-38-160 ~]$
```