Autores: Anselmo Soethe Nurnberg Junior
Ramon Dutra Miranda

O trabalho consiste em um programa que cria threads, utilizando a biblioteca pthread, para concorrer por uma variável global "sharedVar", ele verifica o id da thread utilizando pthread_self() e o seu retorno no pthread_exit().

Exemplo de saida:

```
Processo main() ID = 5627 PPID = 4833
Thread -1217627280 processo ID = 5627 e PPID = 4833.
Thread -1217627280 recebeu mensagem do main() = 0.
Thread -1217627280: shareVar = 0
Thread -1234412688 processo ID = 5627 e PPID = 4833.
Thread -1217627280: shareVar = 0
Thread -1217627280: shareVar = 4
Thread -1217627280: shareVar = 4
Thread -1217627280 retornará 1859712736
Thread -1226019984 processo ID = 5627 e PPID = 4833.
Thread -1226019984 recebeu mensagem do main() = 1.
Thread -1226019984: shareVar = 4
Thread -1226019984: shareVar = 4
Thread -1226019984: shareVar = 4
Thread -1226019984: shareVar = 4
Thread -1226019984 retornará 1842927328
Thread -1234412688 recebeu mensagem do main() = 2.
Thread -1234412688: shareVar = 4
Thread -1242805392 processo ID = 5627 e PPID = 4833.
Thread -1242805392 recebeu mensagem do main() = 3.
Thread -1242805392: shareVar = 4
Thread -1234412688: shareVar = 4
Thread -1234412688: shareVar = 4
Thread -1234412688: shareVar = 4
Thread -1242805392: shareVar = 4
Thread -1234412688 retornará 1826141920
Thread -1242805392: shareVar = 4
Thread -1242805392: shareVar = 4
Thread -1242805392 retornará 1809356512
main() join com thread -1217627280 valor de retorno -1217627280
main() join com thread -1226019984 valor de retorno -1226019984
main() join com thread -1234412688 valor de retorno -1234412688
main() join com thread -1242805392 valor de retorno -1242805392
```

Para compilar utilizamos "gcc -pthread -o thread lab4_ansejr_ramon.rdm.c" e para execução "./thread"

codigo:

```
#include <stdio.h>
#include <stdlib.h>
#include <signal.h>
#include <sys/types.h>
#include <unistd.h>
#include <pthread.h>  /*  Pthread library */
#define num_thread 4

void *thread_rotina(void *); /* prototipo da rotina */
int sharedVar; /* variável global */

int main (){

        void *pRtnValue; /* ponteiro para void */
        pthread_t tid[num_thread]; /* thread id */
        pthread_t departed; /* término */

        /* strings passados para as threads */
        int iValue[num_thread];
        int i; /* contador de threads */
        sharedVar = 0;

         printf("Processo main() ID = %d PPID = %d \n", getpid(), getppid());

        /* cria threads passando argumentos */
        for (i = 0; i < num_thread; i++){
                pRtnValue = i;
                pthread_create(&tid[i], NULL, thread_rotina, pRtnValue);
        }
        /* muda variável global …. */
        for (i = 0; i < num_thread; i++){
                sharedVar++;
                pthread_yield();
        }
        /* espera término de threads */
        for (i = 0; i < num_thread; i++){
                /* mostra ID da thread com valor de retorno */
                if(pthread_join(tid[i], &departed)){
                        perror("pthread_join() falhou.");
                        exit(1);
        }else{
                        printf("main() join com thread %d valor de retorno %d\n", tid[i], departed);
                }
        }
   return(0);
}
//Esqueleto rotina
void *thread_rotina(void *arg){
        int i =0;
        /* mostra  ID do processo */
        printf("Thread %d processo ID = %d e PPID = %d.\n", pthread_self(), getpid(), getppid());
        /* mostra valor passado */
```

```
        printf("Thread %d recebeu mensagem do main() = %d.\n", pthread_self(), (int)arg);
        for (i = 0; i < num_thread; i++){
        /* mostra valor da variável global */
                printf("Thread %d: shareVar = %d\n", pthread_self(), sharedVar);
                pthread_yield();

        }
        /* mostra valor de retorno */
        printf("Thread %d retornará %d\n",pthread_self(), 2*pthread_self());
        /* término thread com argumento */
        pthread_exit(2*pthread_self());
}
```