Sistemas Operacionais – Prof. Rafael R. Obelheiro

Exercícios — Pthreads

1. Compile e execute o programa `simples.c`, disponível nos exemplos de Pthreads. No Linux, isso pode ser feito com os comandos:

```
$ gcc -Wall -o simples simples.c -pthread
$ ./simples
```

2. Modifique o programa `simples.c` para criar, além das *threads* que executam `PrintHello()`, outras `NUM_THREADS`+3 *threads* que imprimam o quadrado do número recebido como parâmetro. (No total, portanto, o programa deve criar 2*`NUM_THREADS`+3 *threads*.)

3. Modifique o programa `simples.c` para que cada *thread* retorne o quadrado do número recebido como parâmetro. O programa principal deve imprimir a soma dos valores de retorno de todas as *threads*.

4. O programa `conta.c`, disponível nos exemplos de Pthreads, conta o números de elementos pares e de elementos ímpares em um vetor de inteiros. Adapte esse programa `conta.c` para criar duas *threads*, uma para contar os pares e outra para contar os ímpares. Ambas as *threads* devem executar a função `conta()`. O vetor `v` pode ser definido como uma variável global, sendo o único parâmetro da função o tipo de número a ser contado. Execute o programa algumas vezes e veja se há diferença no tempo de execução das duas versões (sequencial e *multithread*).

5. Modifique o programa desenvolvido no exercício anterior de modo que tanto o vetor quanto o tipo de número sejam passados como parâmetro para a função `conta()`.

6. Escreva um programa que conte até 2^{31} (`1<<31`). O programa deve ter N *threads*, e cada *thread* deve contribuir $2^{31}/N$ para o total. A contagem em cada *thread* deve ser em incrementos unitários (i.e., `total_thr++`), e o valor final deve ser retornado para `main()` e somado ao total global. Caso a divisão de 2^{31} por N seja inexata, o resto deve ser acrescido em `main()`, também em incrementos unitários. Verifique se, incrementando N, há variação no tempo de execução do programa.

7. Escreva um programa em C que inicialize um vetor global de N inteiros com valores aleatórios (use a função `random()`), e na sequência crie *nthr threads*. Cada *thread* possui um ID numérico (de 1 a *nthr*), e seu comportamento depende do ID: *threads* com ID par devem encontrar o maior elemento do vetor, e *threads* com ID ímpar devem encontrar o menor elemento. Após terminar de percorrer o vetor, a *thread* deve inserir seu ID no fim de uma fila de *threads* concluídas (dica: use um vetor global com *nthr* elementos, e uma variável para indicar em que posição desse vetor deve ser inserido o próximo ID), e retornar o maior/menor elemento. O programa principal deve verificar que o elemento encontrado por uma *thread* está correto, e imprimir uma mensagem de erro caso contrário. Depois que todas as *threads* tiverem encerrado, o programa principal deve imprimir a ordem em que elas terminaram (de acordo com a fila), conforme o exemplo abaixo:

```
ordem: 3 1 2 4
```

   O valor de *nthr* será o primeiro argumento na linha de comando. Se houver um segundo argumento na linha de comando, este será o valor de N; se apenas um argumento for passado na linha de comando, N terá o valor *default* 10.

8. Considere o código abaixo:

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <pthread.h>

int n;

void f1(void *argp) {
    int a;
    a = 20;
    n += a;
    printf("f1: a=%d\n", a);
}

void f2(void *argp) {
    int a;
    a = 10;
    n -= a;
    printf("f2: a=%d\n", a);
}

int main(void) {
    pthread_t t1, t2;
    int rc;
    n = 0;
    rc = pthread_create(&t1, NULL, (void *)f1, NULL);
    rc = pthread_create(&t2, NULL, (void *)f2, NULL);
    rc = pthread_join(t1, NULL);
    rc = pthread_join(t2, NULL);
    printf("n=%d\n", n);
    return 0;
}
```

O que será impresso pelo código acima?

9. Considere o código abaixo:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  #include <sys/types.h>
5  #include <pthread.h>
6
7  #define MAX 2000
8
9  int n;
10
11 void f1(void *argp) {
12     int i, temp;
13     for (i = 0; i < MAX; i++) {
14         temp = n;
15         temp++;
16         n = temp;
17     }
18 }
19
20 void f2(void *argp) {
21     int i, temp;
22     for (i = 0; i < MAX; i++) {
23         temp = n;
24         temp--;
25         n = temp;
26     }
27 }
28
29 int main(void) {
30     pthread_t t1, t2;
31     int rc;
32     n = 0;
33     rc = pthread_create(&t1, NULL, (void *)f1, NULL);
34     rc = pthread_create(&t2, NULL, (void *)f2, NULL);
35     rc = pthread_join(t1, NULL);
36     rc = pthread_join(t2, NULL);
37     printf("n=%d\n", n);
38     return 0;
39 }
```

Qual será o valor de `n` impresso na linha 37? Caso haja mais de um valor possível, enumere as possibilidades e justifique sua resposta.