Sistemas Operacionais

Alisson Linhares

CAMPINAS,

2021/1

Thread em C/C++.

```
#include <iostream>
#include <pthread.h>
using namespace std;

void *thread_func(void *arg) {
   cout << "Olá thread " << (const char*) arg << endl;
}

int main (int argc, char *argv[]) {
   pthread_t a, b;

   cout << "Criando a thread A" << endl;
   pthread_create(&a, NULL, thread_func, (void*)("A"));
   cout << "Criando a thread B" << endl;
   pthread_create(&b, NULL, thread_func, (void*)("B"));

   pthread_join(a, NULL);
   pthread_join(b, NULL);
   cout << "Programa terminou" << endl;
   return 0;
}
```

Digite “g++ -lpthread main.cpp” para compilar.

Thread em C/C++.

```
#include <iostream>
#include <pthread.h>
using namespace std;

void *thread_func(void *arg) {
   cout << "Olá thread " << (const char*) arg << endl;
}

int main (int argc, char *argv[]) {
   pthread_t a, b;

   cout << "Criando a thread A" << endl;
   pthread_create(&a, NULL, thread_func, (void*)("A"));
   cout << "Criando a thread B" << endl;
   pthread_create(&b, NULL, thread_func, (void*)("B"));

   pthread_join(a, NULL);
   pthread_join(b, NULL);
   cout << "Programa terminou" << endl;
   return 0;
}
```

void *foo(void *arg) é o “main” das thread

Thread em C/C++.

```
#include <iostream>
#include <pthread.h>
using namespace std;

void *thread_func(void *arg) {
   cout << "Olá thread " << (const char*) arg << endl;
}

int main (int argc, char *argv[]) {
   pthread_t a, b;

   cout << "Criando a thread A" << endl;
   pthread_create(&a, NULL, thread_func, (void*)("A"));
   cout << "Criando a thread B" << endl;
   pthread_create(&b, NULL, thread_func, (void*)("B"));

   pthread_join(a, NULL);
   pthread_join(b, NULL);
   cout << "Programa terminou" << endl;
   return 0;
}
```

É um registro que identifica cada thread.
Digite "man pthread" no terminal para mais informações.

Thread em C/C++.

```
#include <iostream>
#include <pthread.h>
using namespace std;

void *thread_func(void *arg) {
   cout << "Olá thread " << (const char*) arg << endl;
}

int main (int argc, char *argv[]) {
   pthread_t a, b;

   cout << "Criando a thread A" << endl;
   pthread_create(&a, NULL, thread_func, (void*)("A"));
   cout << "Criando a thread B" << endl;
   pthread_create(&b, NULL, thread_func, (void*)("B"));

   pthread_join(a, NULL);
   pthread_join(b, NULL);
   cout << "Programa terminou" << endl;
   return 0;
}
```

Join espera pelo fim das threads a e b;

LAB04

Lab04: Brute Force - 1/2

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define TAMANHO_CHAVE 8
static char chave_secreta[TAMANHO_CHAVE];

void GerarChaveSecreta(char s[]) {
   static const char letras[] = "abcdefghijklmnopqrstuvwxyz";
   static const int tam = sizeof(letras) - 1;
   for (int i = 0; i < TAMANHO_CHAVE; ++i)
      s[i] = letras[rand() % tam];
   s[TAMANHO_CHAVE] = 0;
}

bool TesteChaveSecreta(char chave[]) {
   if (strcmp(chave, chave_secreta) == 0) {
      printf("Parabéns, você descobriu a chave secreta: %s\n", chave);
      return true;
   }
   return false;
}
```

Lab04: Brute Force - 2/2

```
// Implemente essa rotina usando threads;
void ProcurarChaveSecreta() {
   char chave[TAMANHO_CHAVE];

   do {
      GerarChaveSecreta(chave);
      printf("Testando chave %s\n", chave);
   } while (!TesteChaveSecreta(chave));
}

int main() {
   GerarChaveSecreta(chave_secreta);
   ProcurarChaveSecreta();
   return 0;
}
```

- Modifique a rotina **ProcurarChaveSecreta** para descobrir a senha da aplicação usando duas ou mais threads. O código deve ser feito da forma mais eficiente possível. **Atenção: disponibilize para máquina virtual mais de um processador. Caso a sua cpu tenha hyper-threading, execute duas threads por core!**