Universidade Federal de Uberlândia
Disciplina: GSI018 – Sistemas Operacionais (SO)
Prática: Ambiente Linux

Threads (GABARITO)

1. Threads

1.a – Na execução do código, uma **thread** é criada.

```
user1@ABS:~/Downloads/Adriano-Material/laboratorio3$ ./exerc1
criar uma thread

OLÁ MUNDO.. BEM VINDO :-)
```

1.b – Duas mensagens são exibidas na tela, uma pelo **printf** do algoritmo principal e outra exibida pelo **printf** do **thread** criado.

2. Esperando por um thread

2 – Durante a execução do código, 10 **threads** são criadas. O segundo parâmetro da chamada **pthread_join** é um ponteiro que receberá o estado de saída do **thread** terminado. Como o valor **NULL** foi utilizado, o estado de saída não é retornado.

3. Passagem de argumentos para o thread

3 – Os valores da variável **i** foram presentados, mas não de maneira sequencial e apresentaram repetições.

```
user1@ABS:~/Downloads/Adriano-Material/laboratorio3$ ./exerc3
valor: 3
Espera a finalização das threads criadas
valor: 0
valor: 1
valor: 0
valor: 2
valor: 0
valor: 0
valor: 2
valor: 0
valor: 0
```

4. Thread vs fork

4.a – O tempo médio foi de 0,405 para **NTHREADS** = 500.

4.b – O tempo médio foi de 4,096 para **NTHREADS** = 5000 e 37,565 para **NTHREADS** = 50000.

4.c - Para trabalhar com a função fork, você deve adaptar o código para criar processos. Segue um exemplo de código a ser usado para medir essa informação:

```
1. #define NFORKS 5
2.
3. int main(int argc, char *argv[]) {
4.  int i;
5.
6.  for(int i=0;i<NFORKS; i++) {
7.   if(fork() == 0) {
```

```
8.    printf("[filho] pid %d do [pai] pid %d\n",getpid(),getppid());
9.    exit(0);
10.      }
11.    }
12.    for(int i=0;i<NFORKS;i++)
13.     wait(NULL);
14.
15.    }
```

5 – A função **adjustX** serve para modificar o valor da variável global **x**. Uma thread incrementa o valor em 1 enquanto o outro decrementa em -1. Irá imprimir os threads com valores 1 e -1 relacionados a execução. Nesse caso será criado apenas um processo com os threads para manipulação da função. Ao se remover o loop infinito da função **main**, pode ser que as threads não sejam executadas pois o programa irá finalizar sem aguardar sua execução.