Exercícios de Sistemas Operacionais I
Pratica de criação de threads com pthreads

1. Crie um programa com uso da biblioteca *pthread* que crie duas *threads*, cada uma delas mostrando uma mensagem diferente oriunda do argumento.

2. Crie um programa com uso da biblioteca *pthread* que crie duas *threads*, a partir de duas funções diferentes. As duas *threads* devem uma mensagem e na linha subsequente o argumento recebido. Uma das *threads* deve sempre mostrar "Rio Grande do Sul" e a outra "Terra de Gauchos e da UPF".

3. Crie um programa com uso da biblioteca *pthread* que reproduza o funcionamento do código abaixo. Lembre-se que agora não são processos, mas *threads* que devem ser criados.

```
main(){
    int f1; /* Id processo filho 1*/
    int f2; /* Id processo filho 2*/
    printf("Alo do pai\n");
    f1= create_process(codigo_do_filho);
    f2= create_process(codigo_do_filho);
    wait_process( f1);
    printf("Filho 1 morreu\n");
    wait_process( f2);
    printf("Filho 2 morreu\n");
    exit();
}
codigo_do_filho(){
    printf("Alo do filho\n");
    exit();
}
```

4. Crie um programa com uso da biblioteca *pthread* que reproduza o funcionamento do código abaixo. Lembre-se que agora não são processos, mas *threads* que devem ser criados.

```
main(){
    int f1, f2, f3;
    printf("Alo do pai\n");
    f1=create_process(codigo_do_filho);
    f2=create_process(codigo_do_filho);
    printf("Filho 1 criado\n");
    wait_process( f1 );
    printf("Filho 1 morreu\n");
    printf("Filho 2 criado\n");
    f3=create_process(codigo_do_filho);
    printf("Filho 3 criado\n");
    wait_process( f3 );
    printf("Filho 3 morreu\n");
    wait_process( f2 );
    printf("Filho 2 morreu\n");
    exit();
}
codigo_do_filho(){
    printf("Alo do filho\n");
    exit();
}
```

5. Crie um programa com uso da biblioteca *pthread* que receba um argumento 'x'. A thread deve retornar o valor da expressão $x^2 + 2x + 5$.

6. Crie um programa com uso da biblioteca pthread que receba um argumento ‘x’. A thread deve retornar o valor da expressão x2 + 2x + 5.

7. Crie um programa que crie N *threads*, sendo N um número recebido via linha de comando (ver códigos exemplo das *system calls*). As *threads* devem ser criadas com base em duas funções (criadas no exercício 2). A metade das *threads* deve ser criada com a primeira função enquanto a outra metade com a segunda função (aproximadamente).

8. Modificar o exemplo `somareduz.c` passado para testar o funcionamento com a linha da função `usleep` em vários locais. Testar com ela dentro da função `soma`, dentro da função `dimi`, em ambas função e na `main`. Testar várias vezes cada uma das posições e verificar quando os resultados saem incorretos. Lembre-se que o valor correto é 100.

9. Modificar o exemplo `somareduz.c` passado de forma que sejam criadas N threads `soma` e N threads `dimi`. O valor de N deve ser passado via linha de comando. A cada criação do par de threads as mesmas devem ser sincronizadas e o valor da variável compartilhada deve ser comparado (o valor correto é 100). Deve-se mostrar o percentual das execuções em que o valor da variável final ficou incorreto. Testar com e sem o *usleep* dentro das threads.

10. Ajustar o código do exemplo `somareduz.c` passado de forma que, independente da quantidade de vezes ou da inserção de *usleep* nas *threads*, o valor final da variável compartilhada seja preservado. Pode-se utilizar qualquer função de sincronização das *threads* disponíveis.

11. Implemente um programa que simule o produtor consumidor utilizando *threads*. O produtor deve gerar um número aleatório par, menor que 1000 e armazenar numa variável compartilhada. O consumidor deve obter o valor gerado aleatoriamente pelo produtor e mostrar o valor gerado. Logo após mostrar, o consumidor deve transformar o número em um valor ímpar somando um valor. Lembre-se o consumidor somente pode mostrar número pares e não testar valores.

12. Modificar o programa do produtor consumidor criado no exercício 11 de modo que o produtor coloque na variável compartilhada apenas números não repetidos.