Lab 1

Nome: Lucca Vieira Rocha | Matrícula: 2011342

Exercício 1

> 💡 Elaborar programa para criar 2 processos hierárquicos (pai e filho) onde é declarada uma variável inteira n com valor inicial = 3. Após o fork() o processo pai faz um loop de 10000 vezes somando 1 à variável n, exibe a mensagem “processo pai, pid=xxxx, n=yyyy” e espera o filho terminar. o processo filho faz um loop de 10000 vezes somando 10 à variável n, exibe a mensagem “processo filho, pid=xxxx, n=yyyy” e termina.

O objetivo desse exercício é aprofundar os temas de `fork()` e `pid` .

```
int main(void)
{
    int n = 3;
    pid_t pid = fork();
    ...
```

logo no início do programa realizo um fork para dividir o programa em pai e filho. O `fork()` retorna na variavel pid. Variável importante para o programa saber se ele é um processo pai ou filho.

```
int main(void)
{
    int n = 3;
    pid_t pid = fork();
    int status;
    if (pid != 0) //Se pid!= 0 o processo é pai
    {
            //código Pai
            waitpid(-1, &status, 0);

    }
    else //Logo esse é o processo filho
    {
        //Códido Filho
        exit(0);
    }
}
```

Agora podemos realizar operações distintas dependendo do tipodo processo

```
if (pid != 0)
    {
```

```
        for (int i = 0; i < 10000; i++)
        {
            n++;
        }
        printf("Processo pai, pid=%d, n=%d\n", getpid(), n);
        waitpid(-1, &status, 0);
        printf("%d",status);
    }
    else
    {
        for (int i = 0; i < 10000; i++)
        {
            n = n + 10;
        }
        printf("Processo filho, pid=%d, n=%d", getpid(), n);
        exit(0);
    }
```

Retorno:

```
Processo pai, pid=7695, n=10003
Processo filho, pid=7696, n=100003
```

Repare que os dois valores de n são independentes e ocupam endereços de memória distintos por isso as operações de n realizadas no pai não afeta o valor no filho e vice versa, eles são objetos completamente diferentes depois do `fork()` .