INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA

DA PARAÍBA

RENNYSON CAVALCANTE SOARES

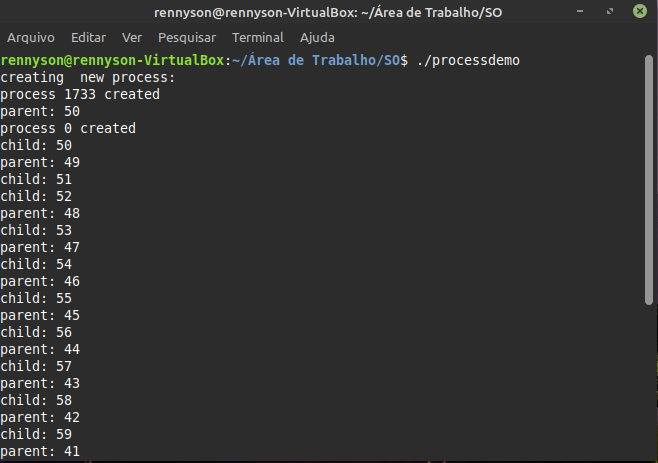
**TÍTULO: Processos e Thread**

Campina Grande-PB

2021

1. **Descreva a saída e explique por que ela é dessa forma?**

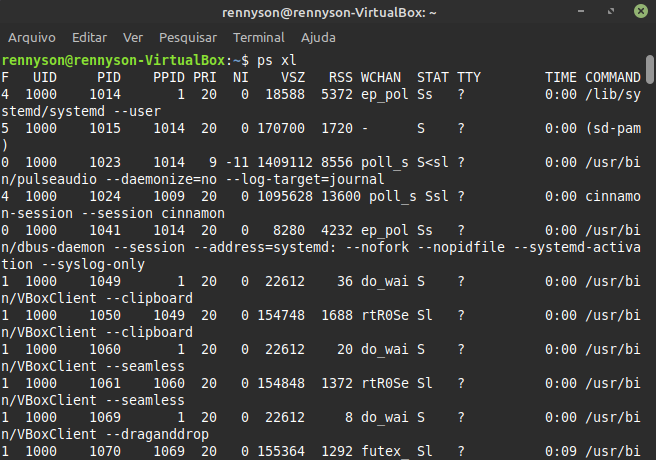
* Rodando código:

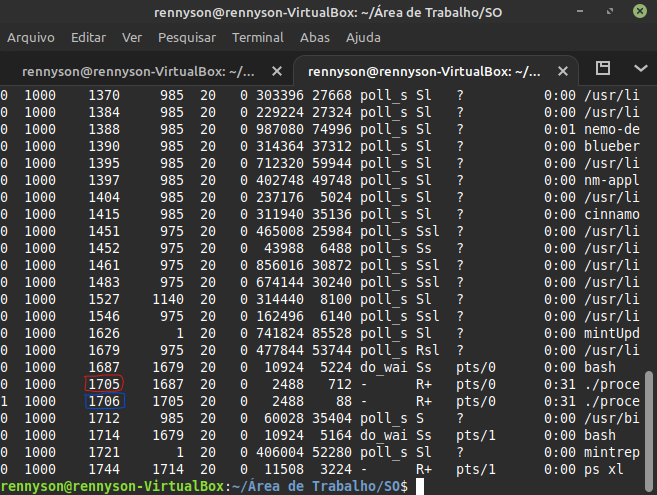


* **A saída ela mostra um processo no meu sistema operacional rodando no momento que realizei a execução, o fork() ele executa a função de cópia ou seja eu continuo com o meu processo rodando pai , mas, ao mesmo tempo eu consigo criar um novo processo filho, o fork no processo Pai ele irá me retornar o PID do processo filho, trazendo uma ideia de alternância entre os processos, o filho executa o processo e em seguida o pai continua. No código é possível perceber que se o processo filho estiver sendo executado ele acrescenta +1 ao filho, e quando a variável “c” for diferente de zero ele entrará no processo pai ele diminuirá -1 ao pai.**

1. **Qual o processo pai e qual o processo filho?**

* Rodando o Código:

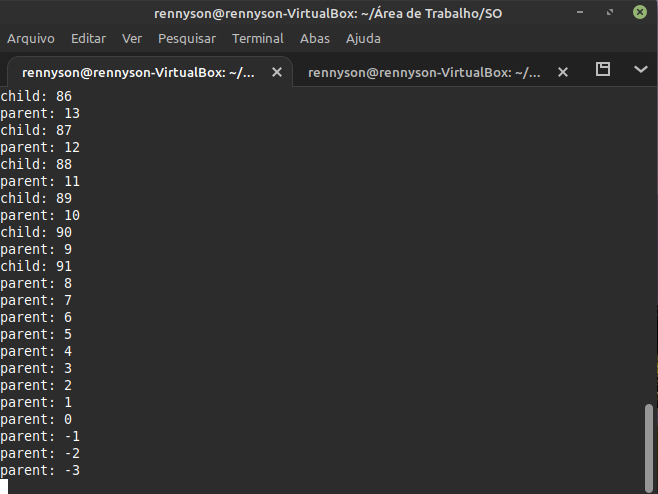


****

* **O PID é o processo pai, e PPID é o processo filho, pois o PID não pode ser único para dois ou mais processos ao mesmo tempo, então o sistema unix precisa que esse processo já existente se duplique, fazendo com que a cópia seja atribuída a uma nova atividade ou seja o processo que está sendo copiado é o pai(o círculo vermelho) , e a nova cópia é o processo filho(círculo azul) chamado de PPID.**

1. **Use o comando "kill -9 PID" para matar o processo filho. O que aconteceu?**

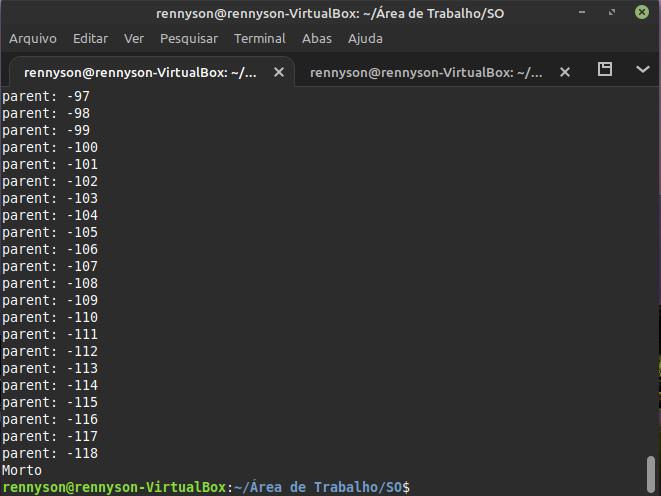
* Rodando o Código:

****

* **o “Kill -9 PID” no filho ele finaliza o processo filho, fazendo com que assim o mate, e o processo pai(parent) continue funcionando, mas o filho ficou em uma zona zumbi.**

1. **Use o comando "kill -9 PID" para matar o processo pai. O que aconteceu?**

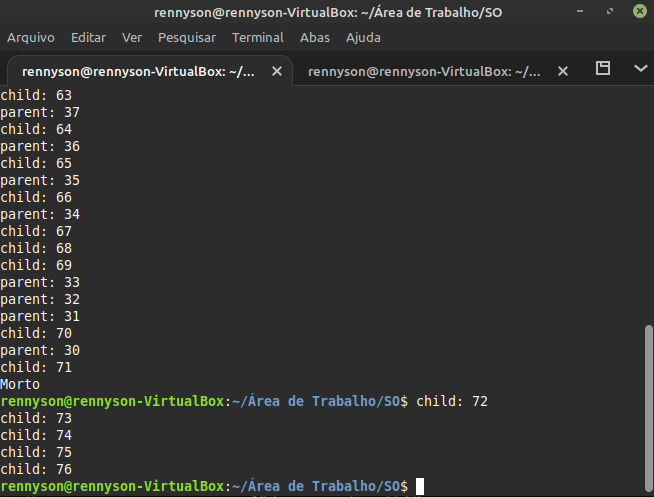
* Rodando o Código:

****

* **O "Kill -9 PID” no pai ele encerra o programa, de maneira forçada fazendo com que assim todo processo seja encerrado, na saída mostra que o processo foi morto e no “ps xl” ele não irá mais aparecer.**

1. **Rode o programa novamente. Identifique e mate o processo pai primeiro em seguida o filho. O que aconteceu?**

* Rodando o código:



* **Matando primeiro o pai , o processo filho “child” irá continuar funcionando e o seu status mudará de “R+” para “R” e em seguida matando o processo filho ele irá encerrar o programa.**

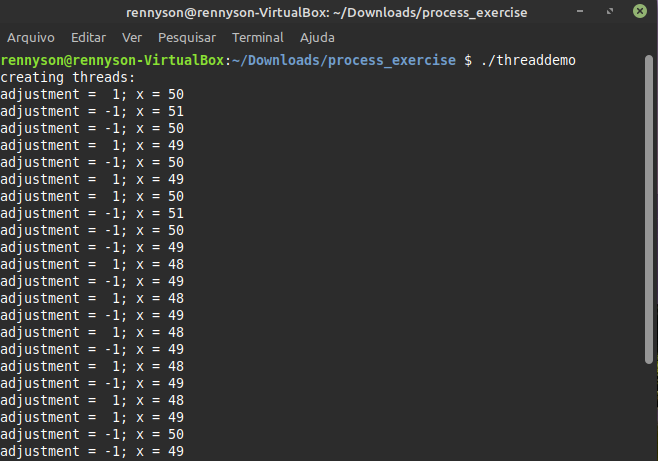
**6.Faz diferença matar o pai ou o filho antes?**

* **sim, realizando primeiro o processo de matar o filho, ele mesmo finalizando o programa , quando verificamos o status dele com o “ps xl” ele está como Z+ ou seja está na zona zumbi que significa que o programa não tá mais funcionando, porém ele ainda continua consumindo recursos do meu computador. Por outro lado, realizando o processo de matar o pai ele encerra o programa de uma vez fazendo com que assim não se possa ter mais arquivos zumbi.**

**7. Rode o programa. O que ele faz? Qual a diferença dele para o programa**

**processdemo.c?**

* Rodando o código:



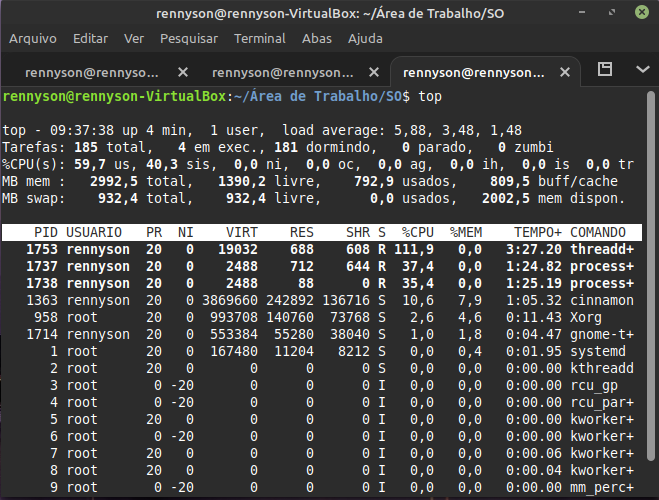
* **A saída mostra o compartilhamento da mesma variável, ou seja , quando um incrementa o outro decrementa.**

**8.Qual a diferença de velocidade de saída (medido em linhas por**

**segundo) comparado a processdemo? Quem é mais rápido? Você tem**

**uma ideia do porquê?**

* Rodando o código:



* **É perceptível que o threaddemo possui uma velocidade maior, pois os threads compartilham o mesmo tempo de fatiamento do processo. Enquanto, o processdemo está utilizando dois processos diferentes, cada um com seu tempo de fatiamento.**

**9.Investigue o efeito de remover o loop infinito no fim do main(). O que**

**acontece? Por que?**

* **Sem o loop ele apenas realiza a criação dos thread, mas não da continuidade por conta que o loop fazia o codigo funcionar e assim mantinha os threads em execução.**

**10.Modifique o programa threaddemo.c para ele fazer a mesma coisa que**

**processdemo.c**

* Rodando o código:

