Atividade 2 de Lab de SO: fork e exec

Aluno: David Galhego || Matrícula: 202304940012

Exercícios para casa:

2° Resposta:

O programa criado por mim pede ao usuário um valor n de processos para serem criados, onde 1 será o processo pai e n-1 será os os processos filhos, onde cada um usa a função “execl(‘./hello’,’./hello’, NULL)”, chamando um arquivo executável hello (criado por meio do make all, que contém o roteiros de compilações e execuções que o programa necessita), o programa hello apenas printa “hello, world” na tela e é finalizado, o processo pai espera que todos os processos filhos sejam finalizados e depois executa ele mesmo a função execl, totalizando n-1 processos filhos e 1 processo pai e um número n de “hello,world” escritos na tela.

O limite de processos que minha máquina minix foi capaz de criar foi de 220, a partir de 240 o processo não é finalizado, possivelmente por limitações de hardware e software de uma máquina Minix.

-> **Perguntas respondidas:**

1. Por que o multiprogramação é fundamental para a operação de um sistema operacional moderno?

**R=É fundamental pois em caso contrário um processo, seja de um programa externo ou nativo, iria monopolizar o processamento do computador, fazendo ele se tornar inoperante durante um período em que possivelmente não necessitaria de toda capacidade de processamento do computador.**

2. Quais são os três estados principais em que um processo pode estar? Descreva sucintamente o significado de cada um.

**R=São eles: estado bloqueado, que impede um processo de ser executado até que algum evento externo ocorra, estado de execução,onde processo está utilizando a cpu para ser desenvolvido, estado pronto, onde o processo já possui um executável para ser utilizado, apenas parado até que seja necessário, possibilitando a cpu de trabalhar com outros processos.**

3. Em todos os computadores atuais, pelo menos parte das rotinas de tratamento de interrupção é escrita em linguagem assembly. Por quê?

**R= É necessário pois essas ações de interrupção utilizam muito o hardware, modificando a forma que ele está atuando, então é necessário uma linguagem de baixo nível para fazer essa interrupção.**

4. Qual é a diferença fundamental entre um processo e uma thread?

**R=O processo possui memória e recursos próprios, enquanto as threads executam “ramificações” desse projeto, indo por diferentes caminhos, mas utilizando a mesma memória e recursos, além disso, os processos são subordinados e gerenciados pelo sistema operacional, enquando as threads é uma unidade de execução dentro do processo,sendo gerenciada pelo processo e não pelo SO.**

5. Em um sistema com threads, existe normalmente uma pilha por thread ou uma pilha por processo? Explique.

**R= Existe uma pilha por thread, pois como cada thread é uma “ramificação” de um processo, ela necessita ter sua própria pilha para contexto, onde uma thread não afeta a outra, além disso serve como segurança no uso da memória e possibilita um maior desempenho para o SO, pois cada thread utiliza apenas a memória que necessita.**

6. O que é uma condição de corrida?

**R=É um problema que pode ocorrer com threads em um sistema multitarefa, onde 2 ou mais threads trabalham com o mesmo dado (lendo ou modificando) sem a devida sincronização, podendo acarretar em erros inconsistentes e imprevisíveis, como dados corrompidos (se uma thread modifica um valor enquanto outra o lê), perda de dados (caso um dado seja modificado ao mesmo tempo por mais de uma thread) ou dados incorretos (por exemplo ler um dado que ainda está sendo modificado)**

7. Dê um exemplo de condição de corrida que poderia ocorrer na compra de passagens aéreas por duas pessoas que querem viajar juntas.

**R= Se essas pessoas verificam a disponibilidade de 2 passagens áreas ao mesmo tempo e em computadores separados, se as 2 iniciarem o processo de compra ao mesmo tempo é capaz do processo de uma falha informando que as duas passagens passaram a não está mais disponíveis no meio do processo de compra ou ainda comprar passagens que não existem na prática.**

3° Parte: Perguntas restantes:

2 dúvidas que restaram foram: É possível fazer com que os processos criados pelo meu código, através dos forks, imprimam o hello world de forma ordenada? por exemplo “processo filho criado, id: XXXX \n Hello world!” pois no meu programa o hello word aparece apenas depois de todos os processos filhos terem sido criados.

A outra dúvida é: Como pode ser feito o controle da visualização das passagens disponíveis do exemplo da pergunta 7? Pois muitas pessoas podem está acessando a disponibilidade das passagens e se ficar apenas disponível apenas para uma pessoa por vez, pode ser ineficaz.