Lista 01 - Sistemas Operacionais

September 14, 2020

Capítulo 01

Questão 1.1)

Uma das grande diferenças entre estações de trabalho e computadores pessoais se dá pela finalidade do S.O. Em PCs, o S.O se preocupa mais em entregar desempenho juntamente com uma interface simples (fácil de utilizar), não se preocupando tanto assim em gerenciar os recursos, visto que teremos um usuário operando o sistema por vez. Já em estações de trabalho, o S.O visa gerenciar melhor os recursos entre os usuários garantindo também usabilidade individual. Estações de trabalho são mais favoráveis em cenários onde o compartilhamento de recursos leve à uso mais eficiente dos mesmos e principalmente aonde o custo associado aos sistemas é elevado.

Questão 1.4)

Timesharing systems serão mais béneficos para o usuário quando estes estiverem sendo pouco utilizados por outros usuários, visto que, esses sistemas possuem muito mais poder computacional do que PCs ou single-user workstation por precisar suportar uma carga de trabalho maior.

Questão 1.8)

Podemos utilizar um bit de modalidade adicionado no hardware, realizando a distinção entre processos do S.O e do usuário, por exemplo.

Questão 1.14)

- Sistemas Uniprocessadores: garantir a coerência de cache em sistemas desse tipo não é necessária visto que apenas teremos um único processo sendo executado por vez.
- Sistemas Multiprocessadores: cada processador terá os seus próprios registradores e cache. Quando o dado é lido do disco magnético pela CPU1, modificado e armazenado no registrador e cache do mesmo, a CPU2 pode ler o mesmo dado não atualizado no disco magnético, visto que a CPU1 ainda não sobreescreveu o valor no disco. Sendo assim, CPU2 deve ser capaz de atualizar o dado quando CPU1 atualizar também.

- Sistemas Distribuídos: Ainda mais complexo que em sistemas multiprocessadores, visto que, o dado pode estar em diferentes computadores sendo atualizado ao mesmo tempo. Com isso, devemos garantir que a atualização em um acesso deve atualizar as dos demais.

Questão 1.22)

Em sistemas multiprocessadores simétricos cada processador executa todas as tarefas do S.O; já em sistemas multiprocessadores assimétricos temos um conceito de "master-slave", onde um processador delega tarefas do S.O à outros processadores mais específicos (GPU, por exemplo). Três vantagens de sistemas mutiprocessadores são aumento de throughput, aumento de confiabilidade e economia de escala; já uma desvantagem é que a taxa de aumento de velocidade quando se usa N processadores não é N, mas sim menor que N por conta do overhead associado para manter corretamente a funcionalidade do sistema. (Além de serem mais complexos em hardware e software do que um sistema uniprocessador)

Questão 1.24)

Interrupções são disparadas pelo hardware para enviar um sinal para a CPU, de I/O por exemplo. Após a interrupção ser devidamente tratada o fluxo retoma de onde parou. Uma trap são interrupções geradas por software, podendo ser geradas intencionalmente por programas de usuário para fazer chamadas de sistema (system calls), por exemplo.

Questão 1.25)

Suponha o caso onde a CPU0 e CPU1 leia o mesmo valor A da memória. Caso a CPU0 atualize o valor de A, essa atualização apenas se dará no cache de CPU0, ou seja, o valor de A para a CPU1 estará desatualizado.

Capítulo 02

Questão 2.1)

1. Criar e excluir arquivos
2. Procurar por arquivos específicos
3. Abrir e fechar arquivos
4. Ler e gravar arquivos
5. Listar informações de arquivos

Questão 2.6)

Sim, nosso TP1 dessa matéria que é desenvolver um shell básico utilizando system calls.

Questão 2.10)

Um sistema projetado em camadas possui uma implementação e depuração mais simples, visto que, a i-ésima camada depende apenas das funções e serviços das camadas anteriores. Uma desvantagem dessa abordagem é o overhead associado na mudança de uma camada mais externa para uma mais interna ao se fazer uma chamada de sistema por exemplo.

Questão 2.17)

Uma semelhança entre a abordagem modular e a de camadas é que cada seção do é bem definida e protegida; já algo diferente é que na abordagem modular, qualquer módulo pode chamar diretamente outro, sem precisar passar por camadas.

Questão 2.19)

Uma vantagem de usar a mesma interface de sistema para manipular arquivos e dispositivos é que a implementação se torna mais simples. Já uma desvantagem é de que, por ser genérica, a interface não aproveita as especificidades que cada um pode oferecer, resultando assim em perda de performance ou funcionalidade.

Questão 2.21)

Como um firmware é relativamente caro e no geral, apenas pequenos montantes ficam disponíveis. Com isso, é comum que sistemas operacionais pequenos sejam armazenados na ROM; enquanto que sistemas operacionais maiores sejam armazenados em disco, onde o seu carregador bootstrap, para inicializar o sistema, seja armazenado em firmware.

Capítulo 03

Questão 3.2)

Se um mecanismo RPC não garantir a semântica de "no máximo uma vez" ou "exatamente uma vez", esse servidor não terá a garantia de que uma requisição não será processada mais de uma vez. Em alguns casos, a falta dessa semântica não tem problema, por exemplo em um banco quando requisitamos o nome ou endereço de uma pessoa dado sua conta bancária. (Note que para depósito e saque isso seria um péssimo problema)

Questão 3.3)

Não, pois o cliente nunca receberá o ACK daquela requisição feita. Com isso, se o cliente enviar novamente uma requisição ao servidor, este irá receber um RPC duplicado, fazendo com que seu processo não seja executado. (It is important to note that the server must send a second ACK back to the client to inform the client the RPC has been performed.)

Questão 3.6)

No primeiro caso o overhead associado a mudança será menor, pois basta redirecionarmos o ponteiro para o registrador que contém o novo contexto; já para o segundo, o processo irá demorar mais, visto que precisamos escolher um dos registradores ocupados, guardar seu valor na memória e usar este para armazenar o novo contexto.

Questão 3.9)

- Escalonador de curto prazo (de CPU): dado os processos na fila de pronto ele escalona aqueles que usarão a CPU. Por ser chamado com maior frequência, sua execução deve ser rápida, já que seus processos duram por menos tempo.
- Escalonador de longo prazo (de jobs): ele escolhe quais processos do poll deverá entrar na fila de pronto. Por ser chamado com menor frequência, sua execução não precisa ser rápida, já que seus processos duram por mais tempo.
- Escalonador de médio prazo: vantajoso quando queremos retirar um processo da memória e reintroduzí-lo para a retomada da execução. Usado, por exemplo, quando um processo consome muita memória.

Questão 3.10)

```
#include  <stdio.h>
#include <unistd.h>
int main(){
/* fork a child process*/
fork();
/* fork another child process*/
fork();
/*and fork another*/
fork();
return 0;
}
```

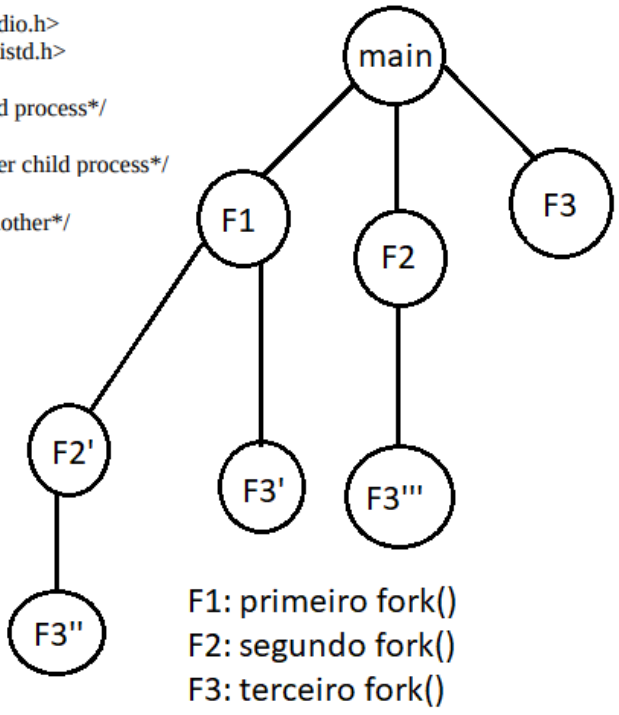

Figure 1: Árvore de criação dos processos

Baseado na figura acima, podemos ver que foram criados 8 processos incluindo a main.

Capítulo 04

Questão 4.1)

1. Um programa que conta até N, onde N é um valor da ordem de 10^{18}, por exemplo.
2. Um programa que possui muito compartilhamento de dados, causando assim um grande uso de mutexes e semáforos no geral, que são caros.

Questão 4.3)

Threads compartilham variáveis globais e heap, onde cada thread possui seus próprios registradores e pilha.

Questão 4.6)

Duas diferenças entre threads de usuário e de kernel é que a primeira é gerenciada por uma biblioteca à nível de usuário, não envolvendo o kernel, e são mais leves, pois o overhead é limitado ao programa que está sendo executado; já a segunda, é integrada ao escalonador do S.O e são gerenciadas e suportadas pelo kernel, possuindo mais overhead.

Questão 4.10)

Já que uma thread é menor do que um processo, por compartilharem variáveis globais, heap, etc, temos que os recursos usados para se criar uma thread também é menor. Para cada thread precisamos criar um contador de programa, pilha de execução e registradores internos; já para processos precisamos, além destes, criar e inicializar um PCB, que é pesado.

Questão 4.15)

Salvar na memória os valores dos registradores da thread que queremos trocar e ler os registradores da nova thread que queremos executar.