Rick Elch

10,0
=
+1,0

Sistemas Operacionais I
1º Semestre de 2023
1ª Avaliação

Francisco José da Silva e Silva
Departamento de Informática, UFMA

Responda a apenas 4 das questões abaixo

Questão 01 Em um sistema de tempo repartido, como o sistema operacional faz para controlar o tempo de execução de um processo? Quais tarefas devem ser realizadas pelo sistema operacional para que possa substituir o processo que estava em execução na máquina por um outro que tenha sido escolhido pelo escalonador entre os processos prontos para executar?

Questão 02 Explique como funciona uma chamada ao sistema operacional, desde a realização da mesma no código do programa do usuário até o retorno de sua execução pelo sistema operacional.

Questão 03 Explique para que serve e como funcionam as chamadas ao SO `fork()` e `execve()`. O que `fork()` retorna e como este valor retornado é utilizado?

Questão 04 O que são e para que servem threads? Exemplifique o uso de threads em uma aplicação servidora. Quais adaptações deve ser feitas para que se possa ter suporte a threads na Tabela do Processo, utilizada pelo SO para gerenciar a execução de um programa?

Questão 05 Explique como funciona a implementação de Mutexes ilustrada na figura abaixo, utilizada para a resolução do problema de exclusão mútua.

```
mutex_lock:
        TSL REGISTER,MUTEX
        CMP REGISTER,#0
        JZE ok
        CALL thread_yield
        JMP mutex_lock
ok:     RET

mutex_unlock:
        MOVE MUTEX,#0
        RET
```

1 R- Em um sistema de Time Sharing, o sistema operacional controla o tempo de execução de um processo configurando o tempo máximo, que um processo fica na CPU, no relógio (uma recurso de hardware) que gera uma interrupção de relógio quando esse tempo máximo é atingido. 2,5

Para que o sistema operacional possa substituir um processo que estava em execução para outro que está pronto e foi escalonado é necessário que se salve o contexto de execução do processo que está na CPU, isto é, o valor do Contador de Programa, dos registradores e e.t.c na tabela de processo e depois carregue o contexto de execução do processo que foi escalonado. A tabela de processo pode está localizado na memória ou no disco.

2 R- 2,5

1 - No programa do usuário é feito o empilhamento dos valores dos parâmetros da chamada
2 - No programa do usuário é feito a chamada do sistema. chamando uma função da bib do SO
3 - A biblioteca, salva o código da chamada num registrador
4 - A biblioteca, chama a instrução TRAP alterando para o modo núcleo
5 - O núcleo acessa o vetor de interrupções e o código salvo no registrador
6 - O núcleo despacha para o tratador de chamadas do sistema
7 - O núcleo retorna para a biblioteca
8 - A biblioteca retorna para o programa do usuário
9 - O programa do usuário prossegue sua execução

3 R- Tanto o fork() como o execve() estão presente no POSIX.

A chamada fork() é utilizado para criar um novo processo filho como uma cópia exata do processo pai (que fez a chamada fork()) e retorna o PID (Identificador do Processo) do novo processo filho. O PID retornado pode ser utilizado para identificar e diferenciar o processo pai do processo filho permitindo ao processo pai o total controle do processo filho.

A chamada execve() é utilizado para fazer um processo filho executar uma determinada função ou método. Portanto, o execve complementa o fork, pois enquanto o fork cria uma cópia do processo pai, este pode usar o execve para dizer ao processo filho o que executar. 2,5

[illegible] Rep

5 R:

A função mutex_lock é utilizada para travar o mutex e permitir que o programa entre na sua região crítica e funciona da seguinte maneira:

1- TSL REGISTER, MUTEX: Essa instrução copia o valor de MUTEX para o REGISTER e altera o valor de MUTEX para 1.

2- CMP REGISTER, #0: Compara se o antigo valor do MUTEX era 0.

3- JLE OK: Pula para a função OK, se o antigo valor de MUTEX era 0, fazendo o retorno para o programa e permitindo que ele entre na sua região crítica

4- Chh thread_yield: libera o processador para que outro processo ou thread seja executado. (Se o antigo valor do MUTEX era 1)

5- JMP mutex_lock: Chama novamente a função mutex_lock para novamente tentar travar o mutex criando um laço de repetição até conseguir travar o MUTEX.

A função mutex_unlock é utilizada para que um programa destrave o mutex depois que dele sair de sua região crítica e permitir que outros acessem a memória compartilhado:

1- MOVE MUTEX, #0: alterar o valor de mutex para 0 destravando-o

2- RET: Retorna para que o programa prossiga com sua execução