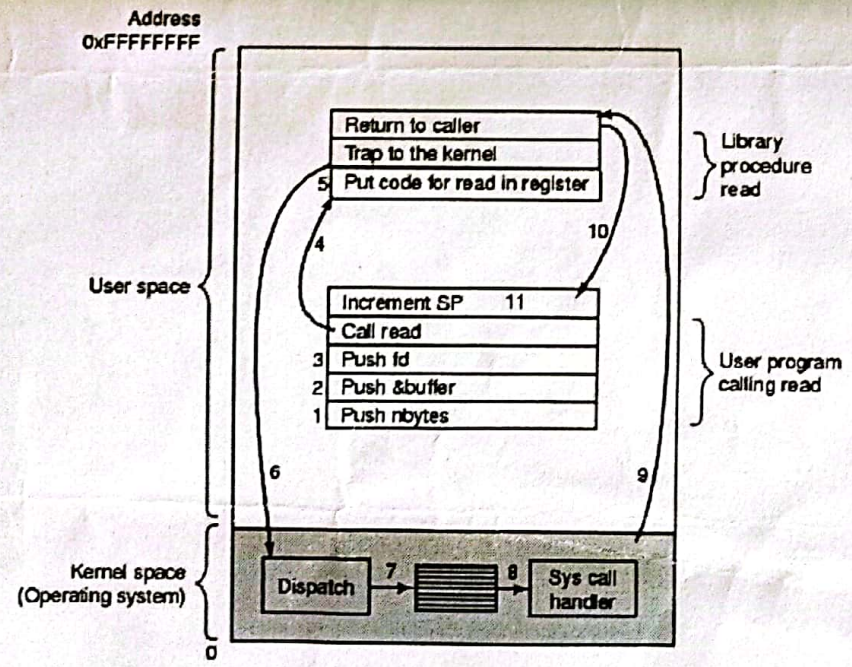
Sistemas Operacionais I
2º Semestre de 2022
1ª Avaliação

Francisco José da Silva e Silva
Departamento de Informática, UFMA

Responda a apenas 4 das questões abaixo

Questão 01 Explique como funciona um sistema um sistema operacional de lote (batch) e um sistema operacional de tempo repartido (time sharing). No sistema de tempo repartido como o sistema operacional faz para controlar o tempo de execução de um processo?

Questão 02 A figura abaixo ilustra os passos para a realização de uma chamada ao sistema operacional. Explique, passo a passo, as etapas necessárias para a execução da chamada ao SO seguindo a numeração apresentada na figura.



Questão 03 Descreva a estrutura Tabela do Processo, utilizada pelo SO para gerenciar a execução de um programa. Se o sistema operacional gerencia as threads instanciadas por um processo, quais adaptações nessa estrutura são necessárias?

Questão 04 O código abaixo resolve o problema da exclusão mútua? Se não resolve, ilustre uma situação em que ele falhe. Quais as desvantagens/limitações deste código?

```
while (TRUE) {
    while (turn != 0)        /* loop */ ;
    critical_region( );
    turn = 1;
    noncritical_region( );
}
```

(a)

```
while (TRUE) {
    while (turn != 1)        /* loop */ ;
    critical_region( );
    turn = 0,
    noncritical_region( );
}
```

(b)

[redacted]

9,0

01. No sistema operacional do tipo lote, no modelo de SO multiprogramming, os processos submetidos para execução são simultaneamente carregados na memória principal e executados um por vez. Caso o processo em execução faça uma chamada ao sistema operacional que o deixe em um estado bloqueado (como uma requisição para usar a impressora), o SO desvia o fluxo de execução para outro processo na memória.

2,5

No sistema operacional de tempo repartido, o SO delega a cada processo na memória a ser executado um tempo pré determinado para que esse processo seja realizado. Ao término desse tempo, o SO escalona outro processo para ser executado. Nesse tipo de SO, para controlar o tempo de execução de cada processo, é definido, no relógio da máquina e de acordo com o escalonador, determinado tempo no qual será feito o processamento dessa atividade, após o término desse tempo, o relógio da máquina envia um sinal de interrupção para o SO e o fluxo de execução é desviado para uma rotina do SO que salva o contexto de execução do processo e escalona outra aplicação para ser realizada.

2,5

02. 1) O processo informa empilha a quantidade de bytes a serem lidos;
2) O processo informa o endereço do buffer a ser lido;
3) O processo informa onde deve ser armazenado o buffer;
4) O processo chama a função "read" na biblioteca linkeditada ao SO;
5) É colocado, no registrador, o código para a função "read";
6) É realizado um sinal de interrupção (trap) para desviar o fluxo de execução para o SO;
7) No modo kernel, a chamada para a função "read" é identificada e será buscado o endereço da sua rotina;

[redacted]

8) Após a localização do endereço, a chamada é direcionada à rotina que é responsável por processar e devolver essa chamada;
9) A rotina devolve o conteúdo requisitado pela chamada para a biblioteca e o fluxo de execução volta para o modo usuário;
10) A biblioteca envia o conteúdo para o processo que realizou a chamada da função "read";
11) O processo continua sua execução a partir da chamada realizada.

03. A Tabela de Processo é uma estrutura que armazena todos os dados relacionados a um processo, como, por exemplo, o Process ID, o espaço de endereçamento, registradores, o contexto de execução... Quando um processo instancia threads e o SO gerencia essas threads, a Tabela passa a armazenar, além dos itens do processo, os registradores, o contexto de execução, a pilha de execução e identificadores, também, para cada thread instanciada.

04. O código apresentado resolve o problema de exclusão mútua em partes, pois ele trabalha com a chamada em espera, ou seja, caso um processo tente entrar na sua região crítica, mas já há um processo acessando essa região, o primeiro processo ficará preso em um loop apertado, testando a disponibilidade para acesso, o que causa gasto do uso do tempo de processamento e pode causar situações como a espera eterna para que possa acessar região crítica, ferindo um dos princípios de tratamento de exclusão mútua. Como no caso onde um processo de baixa prioridade não pode sair da sua região crítica, pois há sempre uma atividade de maior prioridade sendo escalonada.

2,5