Nome: Davi Augusto Neves Leite
RA: 191027383

Resolução – Lista 4 de Sistemas Operacionais II

- **Princípios de Software de E/S**

1) Considerando o software de E/S dividido nas seguintes quatro camadas (de cima para baixo): Software de E/S ao Nível do Usuário, Software de E/S Independente de Dispositivo, *Device Drivers* e Manipuladores de Interrupção; pode-se garantir tanto o principal objetivo de um Software de E/S que é a independência de dispositivo quanto a outros objetivos, como a uniformidade de nomeação e no tratamento de erros.

Especificamente, pode-se definir a ação de cada camada da seguinte maneira (novamente, de cima para baixo):

- Software de E/S ao Nível do Usuário: está associada a interação direta com os processos do usuário, ou seja, as funções de E/S que pertencem ao compilador (conjunto de bibliotecas). Em outras palavras, é nesta camada em que são realizadas as chamadas de E/S e ao *spooling* (comumente associado a fila de impressão).
- Software de E/S Independente de Dispositivo: está associada a tratar os problemas de E/S que são independentes de dispositivos, como nomear uniformemente e proteger os dados. Além disso, possui a função de alocar informações temporárias (*buffers*) de leitura/escrita em dispositivos.
- *Device Drivers*: trata-se da camada mais importante da hierarquia, uma vez que está associada à programação do controlador de dispositivo, ou seja, define “o que” e “como” o controlador deve fazer com o dispositivo de E/S. Além disso, possui a função de checar os status e informar ao sistema sobre qualquer tipo de erro.
- Manipuladores de Interrupção: por fim, a camada mais abaixo que se refere somente a código de máquina possui a principal função de controlar a rotina no momento da interrupção (aciona o hardware para realizar a operação de E/S), por meio do uso de semáforos. Além disso, indica ao *Device Driver* quando a operação de E/S estiver completa.

2) O *Device Driver* deve ser reentrante, em sistemas multiprogramados, para garantir que um ou mais processos possam utilizar um mesmo dispositivo de E/S, sem que haja a necessidade de tornar a UCP ociosa no bloqueio dos mesmos. Em outras palavras, nos sistemas multiprogramados deve ser possível executar mais de um processo concorrentemente, de tal forma que enquanto um processo aguarda pelo resultado da E/S o outro deve ser executado (garantia de interatividade com o usuário). Para que isso seja possível, o acesso aos dispositivos de E/S devem ser independentes de processo, ou seja, um processo não pode controlar inteiramente o dispositivo de E/S até o término da operação solicitada, sendo possível outro processo acessar o mesmo dispositivo e solicitar o que deseja.

3) O *buffer* nada mais é do que uma área de armazenamento temporário de dados. Desta forma, sua utilização é fundamental para os dispositivos de E/S uma vez que cada dispositivo opera com velocidades de leitura e escrita diferente uns dos outros. Por exemplo, um teclado possui velocidade de leitura/escrita de dados em torno de 10 bytes por segundo, ao passo que um mouse possui velocidade de leitura/escrita de dados em torno de 100 bytes por segundo. Tendo isso em vista, é necessário armazenar os dados em *buffer* tanto para serem enviados aos dispositivos de E/S em sua taxa de transferência adequada, quanto para evitar a retenção da memória numa transferência de dados vindas do dispositivo de E/S. Por fim, a camada de Software de E/S Independente de Dispositivo é a responsável por armazenar o *buffer* dos dispositivos de E/S, seja das informações advindas desses dispositivos ou das informações enviadas de um processo para eles.

4) O *spooling* é uma técnica que utiliza uma área em disco como um grande *buffer*. Em outras palavras, utiliza um espaço de armazenamento temporário de dados (*buffer*) para que a UCP não fique em tempo ocioso aguardando a resposta de um dispositivo de I/O que utilizar os dados desse *buffer*. Dessa forma, os programas podem ser executados de maneira concorrente, garantindo a multiprogramação. Atualmente, essa técnica é comumente utilizada pelo S.O. para gerenciar a fila de impressão da seguinte forma: o arquivo a ser impresso é armazenado temporariamente em disco (arquivo de spool) fazendo com que o programa que

solicitou a impressão fique livre para outras atividades (até mesmo solicitar outra impressão, colocada em fila), ou seja, não necessita aguardar o término da impressão. Após a criação do arquivo de *spool*, este é enviado pelo S.O. para uma fila de impressão que será enviado para a impressora, em definitivo.

- **Deadlocks**

5) Um *deadlock* nada mais é do que uma situação onde dois ou mais processos estão aguardando por um evento que só pode ser gerado por algum dos mesmos processos em espera. Em outras palavras, simboliza uma situação de impasse em que os processos ficam impedidos de continuar suas execuções, esperando uns pelos outros com relação a recursos não-preemptivos.
Para ter a ocorrência do *deadlock* são necessárias quatro condições simultâneas, sendo elas:

- Exclusão Mútua: os processos exigem controle exclusivo sobre o recurso que solicitam;
- Retenção e Espera: os processos mantêm diversos recursos alocados enquanto solicitam novos recursos;
- Ausência de Preemptividade: os recursos não podem ser retirados dos processos enquanto estes não finalizarem o uso;
- Espera Circular: formação de uma cadeia circular de processos, cada qual solicitando o recurso alocado ao próximo da cadeia.

6) De acordo com Tanenbaum, existem quatro estratégias para tratar o problema dos *deadlocks*. São elas:

- Algoritmo do Avestruz: trata-se da abordagem mais básica e comum, presente em sistemas como Windows e Linux, que nada mais é do que ignorar o problema, haja visto que a ocorrência do *deadlock* é extremamente rara e sua recuperação é custosa. Ou seja, nessa estratégia o *deadlock* não é tratado, tendo o sistema operacional reiniciado em caso de ocorrência.
- Detecção e Recuperação: consiste em deixar o *deadlock* acontecer, identifica-lo e recuperar o impasse causado. Para tanto, existem algumas abordagens providas neste tipo de estratégia que é baseada no uso de grafos de processos

e recursos. A primeira delas refere-se à recuperação mediante preempção, ou seja, retira a força um recurso (depende de sua natureza) e atribui ao outro processo. A segunda delas refere-se à recuperação mediante retrocesso, ou seja, salva as informações no disco e retira o recurso, ainda que possa causar problemas pois a utilização do disco é um tipo de recurso. Por fim, a terceira delas refere-se à recuperação mediante eliminação de processos que, como o próprio nome sugere, elimina o processo que causou o impasse.

- Prevenção Dinâmica: utilização de procedimentos cuidadosos na alocação de recursos por processos. Ou seja, deve existir algoritmos que permitam que haja uma ordem correta de escalonamento de processos para que todos possam ser executados sem a ocorrência de *deadlock*, o que na prática torna-se inviável uma vez que é difícil prever todos os recursos desejados por um processo.
- Prevenção Estrutural: está associada a negar uma ou mais das quatro condições de ocorrência, haja visto que o *deadlock* ocorre quando as quatro estão presentes simultaneamente. Desta forma, esse método se preocupa em tratar uma das construções durante a estruturação do núcleo do Sistema Operacional, o que se torna inviável já que evitar uma das ocorrências pode causar uma instabilidade na construção do mesmo.

- **Sistemas de Arquivos**

7) Os sistemas de arquivos referem-se a uma abstração da **memória secundária**, enquanto que os espaços de endereçamento de memória (como paginação) referem-se a uma abstração da **memória principal**. Essa abstração refere-se à organização dos dados na memória. Dessa forma, é possível afirmar que os sistemas de arquivos devem ser estruturas **persistentes** e que possam ser **organizados hierarquicamente**, podendo armazenar **grandes quantidades de informações** quando comparada aos espaços de endereçamento de memória, já que estes geralmente não possuem uma grande quantidade de memória física. Diferentemente da memória principal, a memória secundária pode armazenar os dados de maneira indefinida (não volátil) e, portanto, o sistema de arquivos surge para comportar tal função.

8) O Sistema Operacional é responsável por controlar todas as operações envolvendo o sistema de arquivos, organizando o armazenamento secundário de maneira a comportar as seguintes características: modos de acesso, gerenciamento de cabeçalho e mecanismos de integridade. Dessa forma, levando em consideração que um sistema de arquivos é uma abstração da memória secundária, são necessários mecanismos de segurança para o acesso dos arquivos, tendo o Sistema Operacional como responsável tanto pela implementação quanto pela manipulação, uma vez que o acesso a qualquer dispositivo de E/S é realizado pelo núcleo.

9) Para um projeto de Sistema Operacional, um arquivo pode ser estruturado da seguinte maneira:

- Sequência de *Bytes*: representa um conjunto de *bytes* que possa ser interpretado pelo Sistema Operacional. É o mais comum dentre as três abordagens, sendo utilizado por Windows e UNIX. Neste tipo, o processo é quem formata a informação a ser armazenada no arquivo.
- Sequência de Registro: representa uma informação por campo. Essa estrutura era decorrente dos cartões perfurados, os quais possuíam um limite de informações por cartão. Atualmente, essa estrutura é defasada.
- Árvore: representa informações “conectadas” umas nas outras, na forma de árvore computacional. São utilizadas principalmente por servidores, tendo em vista a grande quantidade de informações e sendo necessário o rápido acesso entre elas.

10)

a) Os arquivos regulares são os mais comuns em um Sistema Operacional. Simbolizam àqueles que são criados e utilizados diretamente pelo usuário. Podem ser do tipo texto, binários ou executáveis.

b) Os diretórios são uma forma especial de arquivo cuja função consiste em manter a estrutura hierárquica do sistema de arquivos, sendo definido por um arquivo que contém uma lista de outros arquivos. Em outras palavras, esse tipo de arquivo engloba outros arquivos (e diretórios), de tal forma a organizar e

controlar o sistema de arquivos, uma vez que se pode controlar o acesso a um conjunto de arquivos a partir de seu diretório associado.

c) Os arquivos especiais de caracteres e blocos são tipos de arquivos restritos para modelas os Dispositivos de Entrada e Saída. Ou seja, representam uma interface para um *Device Driver* para que haja a interação entre o software e o dispositivo desejado (por meio das chamadas de sistema).

11) O Controle de Acesso representa uma das etapas de segurança envolvidas em um sistema de arquivos, uma vez que está associado com as operações de escrita, leitura ou execução de determinado arquivo ou conjunto de arquivos. Para tanto, a implementação está baseada em uma Lista de Controle de Acesso (ACL), a qual concede direitos por **usuário** e não por processo, garantindo que um conjunto restrito de usuários possa manipular determinado arquivo ou conjunto de arquivos com base em suas permissões. Em outras palavras, cada arquivo possui restrições associadas a seu uso (ou criação) levando em conta determinado usuário do sistema.

12) Uma possível estruturação do sistema de arquivos, levando em consideração uma partição da memória secundária, pode ser a seguinte:

Nome do Atributo	**Significado do Atributo**
Bloco de Inicialização	Primeiro bloco da partição ativa. Se a partição contém um S.O., carrega-o para a memória principal.
Superbloco	Possui os principais parâmetros do sistema de arquivos, sendo lido na inicialização ou na primeira utilização do sistema de arquivos.
Gerenciamento de Espaço Livre	Utilização do *bitmap* ou lista de ponteiros para representar os endereços livres para alocação.
I-Nodes (UNIX)	Estruturas que escrevem cada arquivo. Utilizado comumente no UNIX.
Diretório-Raiz	Representa o topo da árvore do sistema de arquivos.

Arquivos e Diretórios	Representa outros arquivos e diretórios a partir deste.

13)

a) 1111 1111 1111 0000

b) 1000 0001 1111 0000

c) 1111 1111 1111 1100

d) 1111 1110 0000 1100

14) Técnicas de armazenamento de arquivos

a) A alocação contígua simboliza o armazenamento dos dados em áreas adjacentes dos dispositivos físicos. Em outras palavras, os dados são armazenados em setores consecutivos dos discos rígidos. Existem três tipos de algoritmos para a sua implementação, sendo eles: *First-fit*, em que o primeiro segmento livre com tamanho para alocar o arquivo; *Best-fit*, em que o menor segmento livre disponível com tamanho para alocar o arquivo e; *Worst-fit*, em que o maior segmento é selecionado. Contudo, independente do algoritmo aplicado, essa abordagem possui uma principal desvantagem que é a ocorrência de fragmentação, o que implica na redução da capacidade efetiva de armazenamento. Por outro lado, as operações de leitura e escrita são as mais eficientes possíveis uma vez que os blocos estão próximos uns dos outros, além da manutenção de diretórios se reduzir ao tamanho do arquivo e do setor inicial.

b) A alocação não-contígua e não indexada, com lista ligada de setores, simboliza um tipo de alocação encadeada, isto é, um arquivo pode ser organizado como um conjunto de blocos ligados logicamente no disco, independente de sua localização física. Desta forma, é possível alocar blocos distantes do disco para os arquivos. Neste primeiro tipo, cada setor do disco contém um ponteiro que pode ser utilizado para indicar outro setor, trazendo vantagens como a eliminação da necessidade de compactação e a acomodação fácil de arquivos

de tamanho variados. Por outro lado, as operações de leitura e escrita tendem a ser ineficientes, uma vez que a fragmentação ainda persiste. Além disso, a inexistência de acesso aleatório aos dados e o fato de que um bloco de dados não seja potência de dois (existência de um ponteiro) ocasionam o lento acesso aos arquivos.

c) A alocação não-contígua e indexada, com lista ligada de setores, representa mais uma forma de alocação encadeada, mas dessa vez com a criação de um índice. Esse índice nada mais é do que uma tabela que contém a relação de todos os blocos do dispositivo, sendo que cada entrada associa um ponteiro. Este ponteiro é o mesmo que existe na alocação não indexada discutida no item anterior, mas neste caso ele não está no próprio arquivo e sim em uma tabela especial. Dessa forma, as seguintes vantagens são trazidas nessa abordagem: acomodação de arquivos com tamanhos variados, permissão do acesso randômico e o bloco de dados de um setor seja potência de dois. Por outro lado, a tabela de alocação pode ser muito extensa e ocupar um grande espaço na memória, fora o fato de que a ocorrência de sua perda implica na perda direta do sistema de arquivos. Um exemplo deste tipo é o sistema de arquivos FAT, utilizado pelo MS-DOS e Windows antigos.

d) A alocação não-contígua com a indexação de nós (*I-Nodes*) está baseada na utilização de uma estrutura de dados (nó) que contenha dois princípios básicos: um conjunto de atributos e entradas para endereços de um bloco de dados no disco. A principal diferença e vantagem deste tipo de indexação para o discutido no item anterior está em que, na ocorrência de um arquivo não puder ser armazenado em uma das entradas presentes, é possível alterar os endereços de blocos de dados por endereços de outros *I-Nodes*, podendo ser simples, duplo ou triplo. Em outras palavras, um nó pode indicar até “N” bloco de arquivos. Dessa forma, as principais vantagens consistem em que os arquivos de tamanhos variados são acomodados facilmente, é permitido o acesso randômico, o bloco de dados de um setor persiste em potência de dois e, o diferente da abordagem do item anterior, a perda de um *I-Node* implica na perda apenas do arquivo (e não do sistema de arquivos). Contudo, assim como nas outras abordagens, o problema da fragmentação é persistido.

15) Bloco de 4KB e cada bloco tem endereço de 4 bytes. Ou seja, a quantidade de endereços de um bloco que um bloco contém é: 4KB/4B = 1024 entradas (1K endereços)

Tamanho máximo do arquivo = E.D. + E.I. = E.D. + E.I.S + E.I.D. + E.I.T.

Tamanho máximo do arquivo = (10 * 4KB) + (1K * 4KB) + (1K * 1K * 4KB) + (1K * 1K * 1K * 4KB)

Tamanho máximo do arquivo ~= 4TB