For both fixed and dynamic memory allocation schemes, the operating system must keep list of each memory location noting which are free and which are busy. Then as new jobs come into the system, the free partitions must be allocated.

These partitions may be allocated by 4 ways:

1. First-Fit Memory Allocation
2. Best-Fit Memory Allocation
3. Worst-Fit Memory Allocation
4. Next-Fit Memory Allocation

These are Contiguous memory allocation techniques.

First-Fit Memory Allocation:

This method keeps the free/busy list of jobs organized by memory location, low-ordered to high-ordered memory. In this method, first job claims the first available memory with space more than or equal to it's size. The operating system doesn't search for appropriate partition but just allocate the job to the nearest memory partition available with sufficient size.

Advantages of First-Fit Memory Allocation:

It is fast in processing. As the processor allocates the nearest available memory partition to the job, it is very fast in execution.

Disadvantages of First-Fit Memory Allocation :

It wastes a lot of memory. The processor ignores if the size of partition allocated to the job is very large as compared to the size of job or not. It just allocates the memory. As a result, a lot of memory is wasted and many jobs may not get space in the memory, and would have to wait for another job to complete.

Best-Fit Memory Allocation

This method keeps the free/busy list in order by size – smallest to largest. In this method, the operating system first searches the whole of the memory according to the size of the given job and allocates it to the closest-fitting free partition in the memory, making it able to use memory efficiently. Here the jobs are in the order from smallest job to largest job.

Advantages of Best-Fit Allocation :

Memory Efficient. The operating system allocates the job minimum possible space in the memory, making memory management very efficient. To save memory from getting wasted, it is the best method.

Disadvantages of Best-Fit Allocation :

It is a Slow Process. Checking the whole memory for each job makes the working of the operating system very slow. It takes a lot of time to complete the work.

Worst-Fit Memory Allocation

In this allocation technique, the process traverses the whole memory and always search for the largest hole/partition, and then the process is placed in that hole/ partition. It is a slow process because it has to traverse the entire memory to search the largest hole.

Advantages of Worst-Fit Allocation :

Since this process chooses the largest hole/partition, therefore there will be large internal fragmentation. Now, this internal fragmentation will be quite big so that other small processes can also be placed in that leftover partition.

Disadvantages of Worst-Fit Allocation :

It is a slow process because it traverses all the partitions in the memory and then selects the largest partition among all the partitions, which is a time-consuming process.

What is Next Fit?

The next fit is a modified version of 'first fit'. It begins as the first fit to find a free partition but when called next time it starts searching from where it left off, not from the beginning. This policy makes use of a roving pointer. The pointer moves along the memory chain to search for the next fit. This helps in, to avoid the usage of memory always from the head (beginning) of the free blockchain.

What is its advantage over first fit?

First fit is a straight and fast algorithm, but tends to cut a large portion of free parts into small pieces due to which, processes that need a large portion of memory block would not get anything even if the sum of all small pieces is greater than it required which is so-called external fragmentation problem. Another problem of the first fit is that it tends to allocate memory parts at the beginning of the memory, which may lead to more internal fragments at the beginning. Next fit tries to address this problem by starting the search for the free portion of parts not from the start of the memory, but from where it ends last time.

Next fit is a very fast searching algorithm and is also comparatively faster than First Fit and Best Fit Memory Management Algorithms.

```
Int main(void) {

        printf("0"); fflush(stdout);
        for (int i = 0; i < 2; I++) {
                if (fork() == 0) {
                        bwDelay();
                        printf("%d", 2*i+1); fflush(stdout);
                        bwDelay();
                        printf("%d", 2*i+2); fflush(stdout);
                        exit(0);
                        }
        }
        for (int i = 0; i < 2; i++   ) {

                wait(NULL);
        }
        return 0;
}
```

O deadlock pode ser abordado através de políticas de prevenção em sentido escrito(deadlock prevention), de prevenção em sentido lato (deadlock avoidance) ou de detenção (deadlock detection) . Faça uma análise comparativa destas politicas

Deadlock Prevention: Essa política tenta evitar completamente a ocorrência de deadlock, geralmente através da implementação de restrições adicionais no algoritmo de gerenciamento de recursos. Por exemplo, pode-se exigir que todos os processos solicitem recursos em ordem crescente de número de identificação, ou que todos os recursos sejam solicitados antes de qualquer uso. Essa política é eficaz em evitar deadlocks, mas pode ser difícil de implementar e pode levar a um desperdício de recursos.

Deadlock Avoidance: Essa política tenta evitar a ocorrência de deadlock, mas não impede completamente a ocorrência de deadlock. Em vez disso, ele usa algoritmos para avaliar se uma solicitação de recurso pode levar a um deadlock e, se for o caso, nega a solicitação. Isso requer que o sistema mantenha informações sobre recursos disponíveis e alocações de recursos, e pode ser mais flexível e eficiente do que a prevenção de deadlock.

Deadlock Detection: Essa política detecta a ocorrência de deadlock e, em seguida, toma medidas para resolver o problema. Isso pode ser feito verificando periodicamente se há processos que estão aguardando recursos que nunca serão liberados, ou verificando se há ciclos de dependência de recursos. Uma vez detectado um deadlock, pode-se resolvê-lo matando um ou mais processos, liberando seus recursos ou mudando a prioridade de processos. Essa política é mais fácil de implementar, mas pode ser mais ineficiente e pode causar perda de dados.

Em resumo, a prevenção de deadlock é a melhor maneira de evitar deadlocks, mas pode ser difícil de implementar e pode levar a desperdício de recursos. A evitação de deadlock é uma boa opção se for possível implementar, mas pode ser complexa. A detecção de deadlock é a opção mais fácil de implementar, mas pode ser ineficiente e pode causar perda de dados. A escolha da política dependerá das necessidades específicas do sistema e da capacidade de implementação.

A possibilidade de ocorrência de deadlock pressupõe a satisfação em simultâneo de 4 condições. Diga quais e descreva!

As quatro condições geralmente aceitas como necessárias para que ocorra um deadlock são:

1. Condição de exclusão mutua: Os recursos compartilhados devem ser exclusivos, ou seja, não podem ser usados simultaneamente por mais de um processo.
2. Condição de não-preempção: Um processo não pode ser forçado a liberar um recurso que já possui.
3. Condição de espera circular: Um conjunto de processos deve estar esperando uns pelos outros. Cada processo está esperando por um recurso que é controlado por outro processo.
4. Condição de existência de recursos: Deve haver um número finito de cada tipo de recurso, e esses recursos devem ser alocados de tal forma que possam ser esgotados.
5.

Essas quatro condições são conhecidas como as condições de "Hold and Wait" (segurança), "No preemption" (não-preempção), "Circular Wait" (espera circular) e "Resource Inexistence" (existência de recursos) respectivamente. A ocorrência simultânea dessas condições é o que permite o deadlock.

Em ambiente linux, as threads são frequentemente designadas por processo leves (lightweight processes). Explique porque.

Em sistemas operacionais, como o Linux, as threads são frequentemente designadas como "processos leves" porque elas compartilham a maioria dos recursos de um processo, mas possuem menos overhead de gerenciamento. Isso significa que as threads têm menos espaço em memória e menos tempo de processamento para gerenciar, o que as torna mais leves que os processos.
As threads compartilham o espaço de endereço do processo pai, o que significa que elas podem acessar as mesmas variáveis e recursos de memória. Isso permite que elas se comuniquem facilmente e colaborem para realizar tarefas. Além disso, as threads são geralmente gerenciadas pelo kernel, enquanto os processos são gerenciados pelo sistema operacional, o que significa que as threads têm menos overhead de gerenciamento.

Por essas razões, as threads são frequentemente usadas em sistemas operacionais para melhorar o desempenho e a escalabilidade do aplicativo, especialmente em aplicativos que precisam de muitas tarefas simultâneas.

Caracteriza First Fit e Best Fit numa organização de memoria

First Fit é um algoritmo de alocação de memória que procura o primeiro bloco de memória livre que é suficientemente grande para armazenar um processo. Ele varre a lista de blocos livres da memória, começando pelo início, até encontrar um bloco que atenda às necessidades do processo. Esse bloco é então dividido em duas partes: uma para armazenar o processo e outra para deixar como livre.
Best Fit é um algoritmo de alocação de memória que procura o bloco de memória livre mais apropriado para armazenar um processo. Ele varre a lista de blocos livres da memória, classificando-os por tamanho e escolhendo o bloco com o tamanho mais próximo ao necessário pelo processo. Esse bloco é então dividido em duas partes: uma para armazenar o processo e outra para deixar como livre.
Enquanto First Fit é mais rápido, Best Fit tende a maximizar a utilização da memória e minimizar desperdício de espaço.