**Linux Schedular**

The Linux scheduler is a component of the Linux kernel responsible for managing the execution of processes on a system with multiple tasks or processes. Its primary goal is to efficiently allocate the CPU (Central Processing Unit) time to different tasks, ensuring fair access and optimal utilization of system resources.

Key Words

**Burst Time:**

**Definition:** Burst time is the amount of time a process requires to execute its CPU-bound code without any interruption, such as waiting for I/O (Input/Output) or other events.

**Turnaround Time:**

**Definition:** Turnaround time is the total time taken by a process to complete its execution, starting from the submission of the process to the system until it finishes.

**Waiting Time:**

**Definition:** Waiting time is the total time a process spends waiting in the ready queue, waiting for its turn to execute on the CPU.

1. **First-Come, First-Served (FCFS):**

**Description**: The first process that arrives is the first to be executed.

**Advantages**: Simple and easy to understand.

**Disadvantages**: Can result in poor turnaround time and may lead to the "convoy effect" where shorter processes are delayed by longer ones.

1. **Shortest Job Next (SJN) or Shortest Job First (SJF):**

**Description**: The process with the shortest burst time is scheduled next.

(Burst Time: the time it takes for a process to use the CPU without being interrupted by waiting for I/O or other events)

**Advantages**: Minimizes waiting time and improves turnaround time.

**Disadvantages**: Requires knowledge of the burst time, which may not be known in advance.

1. **Priority -based Scheduling:**

**Description**: Each process is assigned a priority, and the process with the highest priority is scheduled first.

**Advantages**: Allows for priority levels to be assigned based on factors like importance or resource requirements.

**Disadvantages**: Can lead to starvation of lower-priority processes if higher-priority processes are continually arriving.

1. **Round Robin (RR):**

Description: Each process is assigned a fixed time slice (quantum), and the scheduler rotates between processes in a circular fashion.

**Advantages**: Fair distribution of CPU time, suitable for time-sharing systems.

**Disadvantages**: May result in poor performance for long-running processes, and short time slices may lead to high context-switching overhead.

1. **Completely Fair Scheduler (CFS):**

**Description**: Used in the Linux kernel, CFS aims to provide fair distribution of CPU time among processes based on the concept of virtual runtime.

**Advantages**: Fairness in resource allocation.

**Disadvantages**: Complexity in implementation.

**Memory Management**

1. **Memory Allocation:**

Dynamic Memory Allocation: In C, dynamic memory allocation is performed using functions like malloc(), calloc(), realloc(). These functions allocate memory at runtime from the heap.

* malloc(size\_t size): Allocates a specified number of bytes of memory. The content is uninitialized.
* calloc(size\_t num, size\_t size): Allocates memory for an array of elements, each of a specified size. The content is initialized to zero.
* realloc(void \*ptr, size\_t size): Changes the size of the previously allocated memory block.
* int \*arr = (int \*)malloc(5 \* sizeof(int));  // Allocating an array of 5 integers

1. **Memory Deallocation:**

Freeing Memory: The free() function is used to deallocate memory that was previously allocated using malloc(), calloc(), or realloc().

free(arr);  // Deallocating the memory

1. **Memory Access:**

**Pointer Arithmetic:** C allows direct manipulation of memory through pointers. Pointer arithmetic can be used to navigate through arrays and structures.

int \*ptr = arr;

printf("%d", \*(ptr + 2));  // Accessing the third element of the array

1. **Memory Copying:**

memcpy(): The memcpy() function is used to copy a block of memory from one location to another.

int source[] = {1, 2, 3, 4, 5};

int destination[5];

memcpy(destination, source, sizeof(source));

1. **Memory Initialization:**

memset(): The memset() function sets a block of memory to a particular value.

int arr[5];

memset(arr, 0, sizeof(arr));  // Initializes the array to zero