

**Air crafts Avionics using priority scheduling**

OS Project



**Team Members:**

K. Kaushik Srinivas

K.Sushanth

K. Adithi

**ABSTRACT**

The following C code demonstrates a priority-based scheduling algorithm for processes, where each process represents an aircraft task in the context of avionics. In the avionics domain, tasks are characterized by their execution times (burst times) and priorities. The objective is to efficiently schedule these tasks to minimize waiting times and improve overall system performance.

To adapt this scheduling algorithm for aircraft avionics, the burst time and priority inputs can be related to the estimated execution time and criticality of each avionic task, respectively. The program employs a selection sort algorithm to arrange tasks in ascending order based on priority. The waiting time and turnaround time are then calculated for each task, reflecting the time spent waiting in the queue and the total time taken from task submission to completion.

The adapted algorithm serves as a basis for optimizing task scheduling within the avionics system, ensuring that critical tasks with higher priorities are executed promptly. The output provides insights into the performance of the scheduling algorithm, including average waiting time and average turnaround time, metrics crucial for assessing the efficiency of the avionics system.

Please note that when implementing this in an aircraft avionics system, it is essential to consider real-time constraints, safety requirements, and the specific characteristics of avionic tasks.

**Objectives of Aircraft Avionics Systems Using Priority Scheduling:**

1.Enhanced Safety and Reliability:

* Prioritize critical avionics tasks to ensure the safety of the aircraft.
* Reduce the risk of system failures by allocating resources based on priority.

2.Optimized System Performance:

* Improve overall avionics system efficiency by scheduling tasks based on their importance and urgency.
* Ensure that essential functions receive sufficient resources to operate smoothly.

3.Real-time Responsiveness:

* Achieve low-latency responses for time-sensitive avionics applications.
* Guarantee timely execution of critical tasks, such as navigation and collision avoidance.

4.Resource Allocation Management:

* Allocate computational resources judiciously to prevent bottlenecks and optimize system performance.
* Manage processor, memory, and communication bandwidth efficiently for various avionics subsystems.

**Applications of Aircraft Avionics Systems Using Priority Scheduling:**

1.Flight Control Systems:

* Prioritize control algorithms for ailerons, elevators, and rudders to ensure real-time response and stability.
* Allocate resources based on the criticality of flight control tasks during different flight phases.

2.Navigation Systems:

* Schedule GPS and inertial navigation system updates with high priority to maintain accurate position information.
* Ensure timely execution of route planning and trajectory calculations.

3.Communication Systems:

* Prioritize air-to-ground and inter-aircraft communication for critical updates and coordination.
* Optimize data transmission scheduling for reliable and timely exchange of information.

**Algorithm for Aircraft Avionics Systems with Priority Scheduling:**

1.Input:

* Read the total number of flights n.
* For each flight i from 1 to n:
* Read and store the arrival time, departure time, and priority of the flight.

2.Sorting Flights Based on Priority:

* Use the selection sort algorithm to sort the flights based on priority.
* For each flight i from 1 to n:
* Find the flight with the minimum priority among the remaining flights.
* Swap the current flight with the flight having the minimum priority.

3.Print Order of Flights:

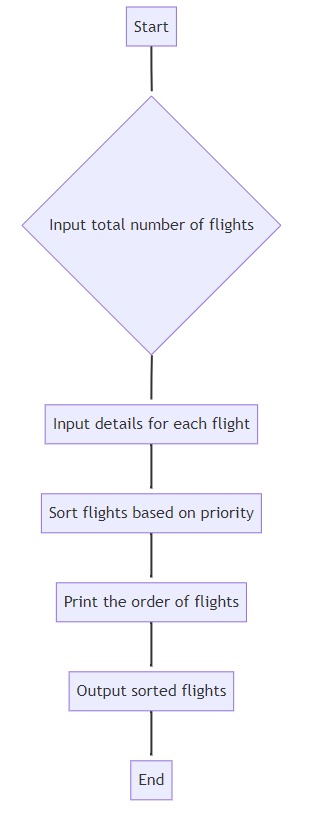
* For each flight i from 1 to n:
* If it is the first flight, print "Flight i goes first."
* Otherwise, print " -> Flight i goes next."

4.Output Sorted Flights:

* Print the table of sorted flights with columns for Flight ID, Arrival Time, Departure Time, and Priority.

5.End

Flowchart:



**Code:**

#include <stdio.h>

// Structure to represent a flight

struct Flight {

int id; // Flight ID

int arrivalTime; // Arrival time

int departureTime; // Departure time

int priority; // Priority based on some criteria (you can customize this)

};

int main() {

struct Flight flights[20];

int n, i, j, total = 0, pos, temp;

// Input the number of flights

printf("Enter the total number of flights: ");

scanf("%d", &n);

// Input details for each flight

printf("\nEnter Arrival Time, Departure Time, and Priority for each flight\n");

for (i = 0; i < n; i++) {

printf("\nFlight %d\n", i + 1);

flights[i].id = i + 1; // Flight ID

printf("Arrival Time: ");

scanf("%d", &flights[i].arrivalTime);

printf("Departure Time: ");

scanf("%d", &flights[i].departureTime);

printf("Priority: ");

scanf("%d", &flights[i].priority);

}

// Sorting flights based on priority using selection sort

for (i = 0; i < n; i++) {

pos = i;

for (j = i + 1; j < n; j++) {

if (flights[j].priority < flights[pos].priority)

pos = j;

}

temp = flights[i].priority;

flights[i].priority = flights[pos].priority;

flights[pos].priority = temp;

temp = flights[i].arrivalTime;

flights[i].arrivalTime = flights[pos].arrivalTime;

flights[pos].arrivalTime = temp;

temp = flights[i].departureTime;

flights[i].departureTime = flights[pos].departureTime;

flights[pos].departureTime = temp;

temp = flights[i].id;

flights[i].id = flights[pos].id;

flights[pos].id = temp;

// Print the order of flights

if (i == 0) {

printf("\nFlight %d goes first", flights[i].id);

} else {

printf(" -> Flight %d goes next", flights[i].id);

}

}

// Output the sorted flights based on priority

printf("\n\nFlight\tArrival Time\tDeparture Time\tPriority");

for (i = 0; i < n; i++) {

printf("\n%d\t%d\t\t%d\t\t%d", flights[i].id, flights[i].arrivalTime,

flights[i].departureTime, flights[i].priority);

}

return 0; }

**Output:**

