Lab-7

|  |
| --- |
| #include <stdio.h>  int current[5][5], maximum\_claim[5][5], available[5];  int allocation[5] = {0, 0, 0, 0, 0};  int maxres[5], running[5], safe = 0;  int counter = 0, i, j, exec, resources, processes;  int main() {  printf("\nEnter number of processes: ");  scanf("%d", &processes);  for (i = 0; i < processes; i++) {  running[i] = 1;  counter++;  }  printf("\nEnter number of resources: ");  scanf("%d", &resources);  printf("\nEnter Claim Vector:\n");  for (i = 0; i < resources; i++) {  scanf("%d", &maxres[i]);  }  printf("\nEnter Allocated Resource Table:\n");  for (i = 0; i < processes; i++) {  for (j = 0; j < resources; j++) {  scanf("%d", &current[i][j]);  }  }  printf("\nEnter Maximum Claim Table:\n");  for (i = 0; i < processes; i++) {  for (j = 0; j < resources; j++) {  scanf("%d", &maximum\_claim[i][j]);  }  }  printf("\nThe Claim Vector is:");  for (i = 0; i < resources; i++) {  printf("\t%d", maxres[i]);  }  printf("\nThe Allocated Resource Table:\n");  for (i = 0; i < processes; i++) {  for (j = 0; j < resources; j++) {  printf("\t%d", current[i][j]);  }  printf("\n");  }  printf("\nThe Maximum Claim Table:\n");  for (i = 0; i < processes; i++) {  for (j = 0; j < resources; j++) {  printf("\t%d", maximum\_claim[i][j]);  }  printf("\n");  }  // Calculate allocated resources  for (i = 0; i < processes; i++) {  for (j = 0; j < resources; j++) {  allocation[j] += current[i][j];  }  }  printf("\nAllocated resources:");  for (i = 0; i < resources; i++) {  printf("\t%d", allocation[i]);  }  // Calculate available resources  for (i = 0; i < resources; i++) {  available[i] = maxres[i] - allocation[i];  }  printf("\nAvailable resources:");  for (i = 0; i < resources; i++) {  printf("\t%d", available[i]);  }  printf("\n");  // Banker's algorithm  while (counter != 0) {  safe = 0;  for (i = 0; i < processes; i++) {  if (running[i]) {  exec = 1;  for (j = 0; j < resources; j++) {  if (maximum\_claim[i][j] - current[i][j] > available[j]) {  exec = 0;  break;  }  }  if (exec) {  printf("\nProcess %d is executing", i + 1);  running[i] = 0;  counter--;  safe = 1;  for (j = 0; j < resources; j++) {  available[j] += current[i][j];  }  break;  }  }  }  if (!safe) {  printf("\nThe processes are in unsafe state.\n");  break;  } else {  printf("\nThe process is in a safe state.");  printf("\nAvailable vector:");  for (i = 0; i < resources; i++) {  printf("\t%d", available[i]);  }  printf("\n");  }  }  return 0;    } |



