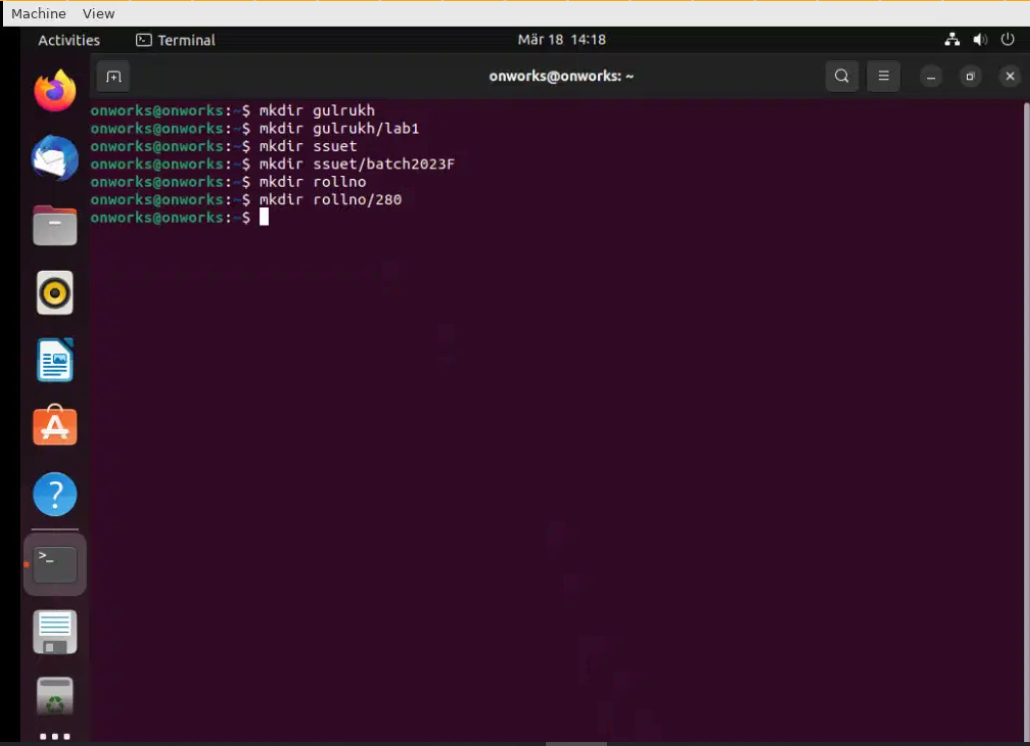
*LAB # 07*

open ended

LAB task # 01

1. *To organize personal academic information in a structured way using the Linux mkdir command. This task helps the student practice directory creation and understand the importance of a well-organized file system.*

******

*LAB task # 02*

*A student is learning about* ***process management*** *in an Operating Systems course. The instructor assigns a task to write a program that demonstrates how a child process is created using the fork() system call in C.*

*The student writes* ***Program06****, where a child process is created from a parent process. Each process then prints:*

* *Its own* ***Process ID (PID)***
* *Its* ***Parent Process ID (PPID)***

*This helps the student observe how a single program can run as* ***two separate processes*** *after the fork() call.*

***Code:***

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <unistd.h>  int main()  {  int pid;  pid=fork( );  if(pid==0)  {  printf("I am the child, my process ID is %d\n",getpid( ));  printf("The child’s parent process ID is %d\n",getppid( ));  }  else  {  printf("I am the parent, my process ID is %d\n",getpid( ));  printf("The parents parent process ID is %d\n",getppid( ));  } } |

Text, letter

Description automatically generated

***Output:***

*LAB task # 03:*

*A student is learning about* ***multithreading*** *in their Operating Systems or Concurrent Programming course. The instructor gives a practical task: implement a program that uses* ***5 separate threads****, each performing the same computational task independently — multiplying two* ***5x5 matrices****.*

*The purpose of this task is to help the student understand how threads can run* ***simultaneously****, sharing CPU time to perform repetitive or parallelizable operations more efficiently.*

*Each thread in the program:*

* *Is created using pthread\_create().*
* *Performs a 5x5 matrix multiplication.*
* *Prints the result along with its thread number.*
* *Exits using pthread\_exit() after completing its computation.*

Graphical user interface, text, application

Description automatically generated**CODE**

Graphical user interface, text, application

Description automatically generated

**Graphical user interface, text, application

Description automatically generated**

Graphical user interface, text, application

Description automatically generated

***Graphical user interface

Description automatically generated with medium confidenceoutput***

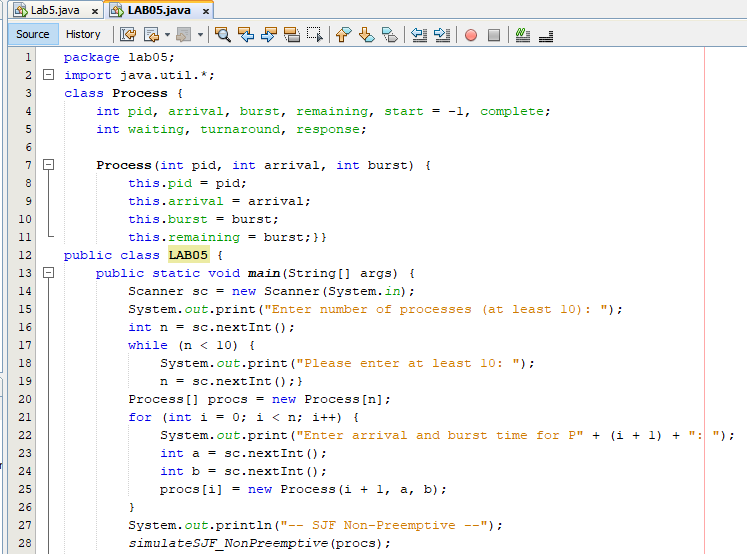
*LAB task # 04*

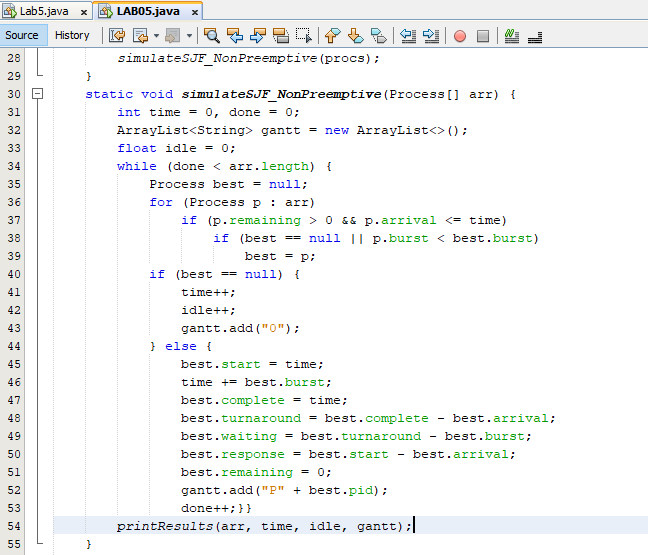
*A student is studying* ***CPU scheduling algorithms*** *as part of an Operating Systems course. The instructor assigns a task to analyze the performance of the* ***SJF (Shortest Job First)*** *algorithm using both* ***non-preemptive*** *and* ***preemptive (Shortest Remaining Time First)*** *approaches.*

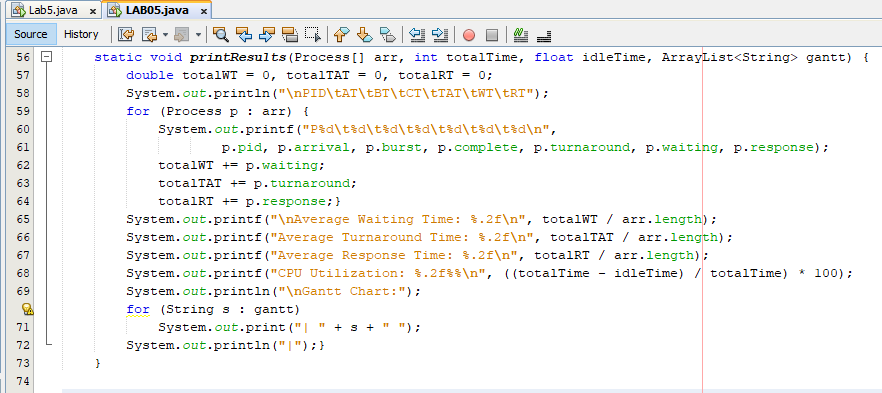
*The program must:*

* *Accept* ***at least 10 CPU burst times*** *as input (e.g., from command-line arguments or user input).*
* *Simulate both SJF (Non-Preemptive) and SJF (Preemptive) scheduling.*
* *Calculate and display the following performance metrics for each algorithm:*
  + ***CPU Utilization***
  + ***Average Waiting Time (AWT)***
  + ***Average Turnaround Time (ATAT)***
  + ***Average Response Time (ART)***
* *Print a* ***text-based Gantt Chart*** *to visualize the scheduling order of the processes.*

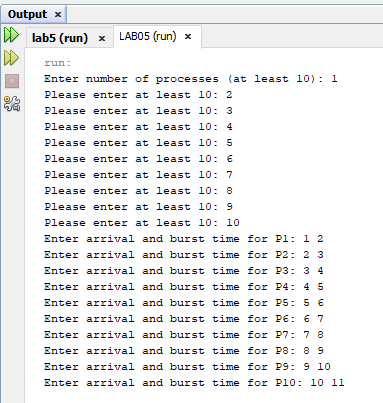
***CODE:***

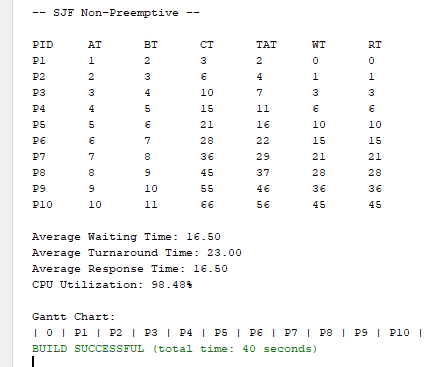
**

**

**

***OUTPUT:***

**

**