**Manarat International University**



Lab Reports: [2, 3, 4] on Operating System

**Course Title: Operating Systems**

Course Code- CSE 310

**Prepared for-**

**Md. Zahurul Haque**

Assistant Professor,

Dept. of Computer Science & Engineering

Manarat International University

**Prepared by-**

**MD. Maksudur Rahman**

ID no: 2258CSE00969

58th Batch, Summer

**Date: 14 September, 2024**

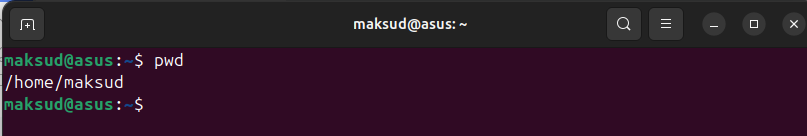
Department of Computer Science & Engineering

**Faculty of Engineering and Technology**

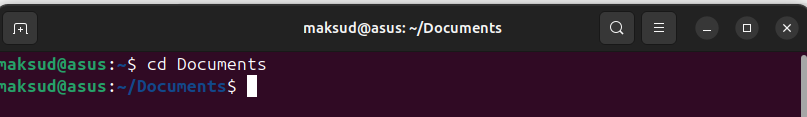
**Manarat International University**

**Lab Report-02 (Basic Commands of Linux)**

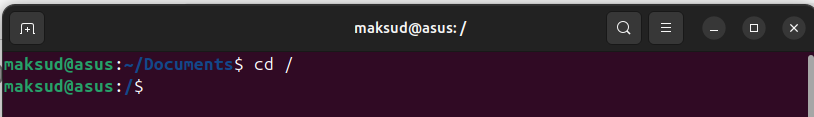
1. See the current directory path we can use “**pwd**”:



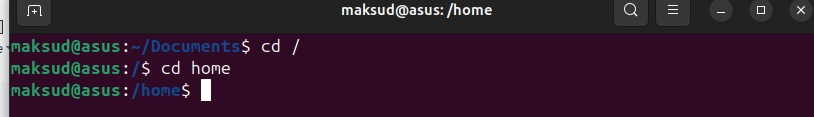
1. Changing the working Directory using “**cd[folder name]**” :



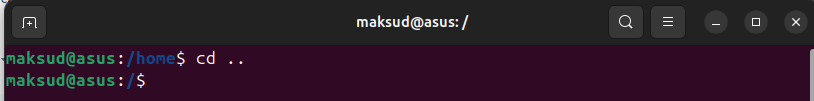
1. Changing the root directory using “**/**” after cd:



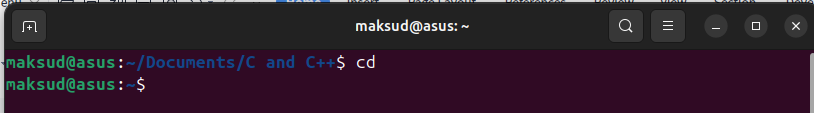
1. Move from root directory to “home” directory using “**$ cd home**” command:



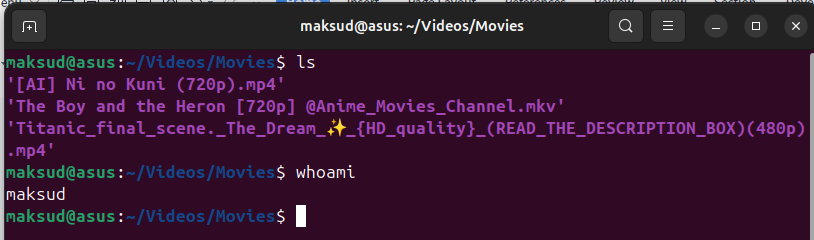
1. Go up to parent directory, in this case back to “**/ ..**” command:



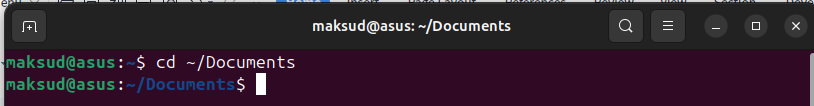
1. Quick shortcut to visit home directory using “**$ cd**” command:



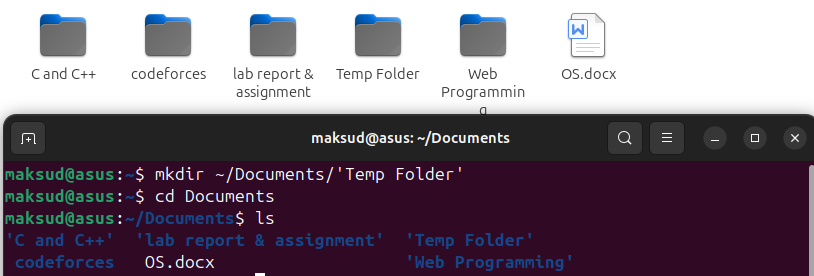
1. Anywhere to home directory using “**$whoami**” command:



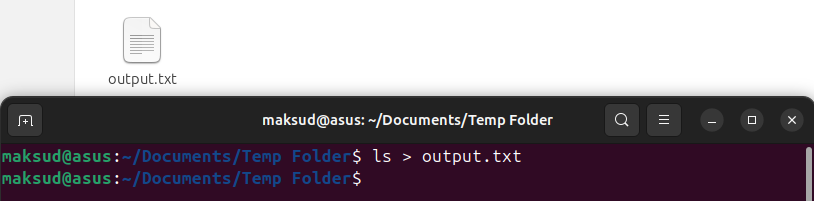
1. Access child folder using “**$ cd ~/**[**folder name**]” command:



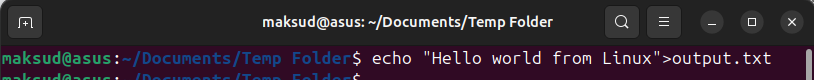
1. Folder creation using “**mkdir”** (make directory) command:



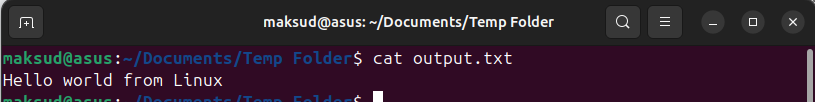
1. Creating a text file using “**$ ls > output.txt**” command:



1. Add content to “output.txt” file using “**$ echo “**[text i want to add]**” > output.txt**” command:

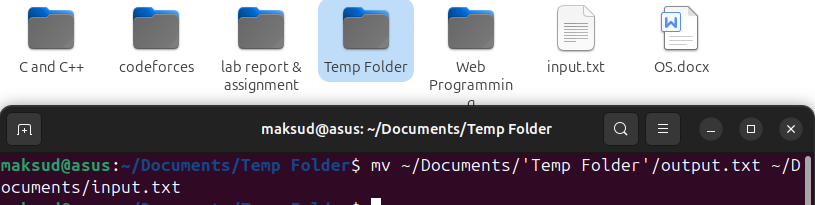


1. We can look thorigh the content using “**cat**” command:

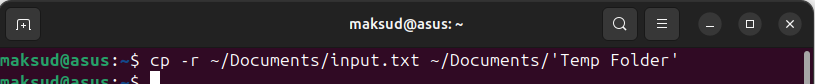


1. Move/Rename the file from one directory to another using “**mv**” command:

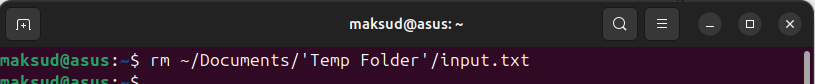
moved from “Temp Folder” to “Documents” and “output.txt” file name changed to “input.txt”



1. To copy the file from one directory to another using “**cp**” command:



1. Delete a file using “**$ rm [file name]**” command:

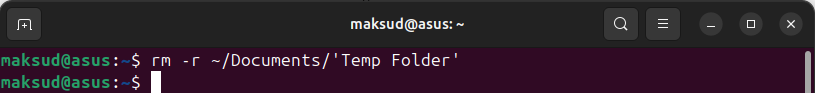


1. Delete a directory or folder using “**rmdir**” command:

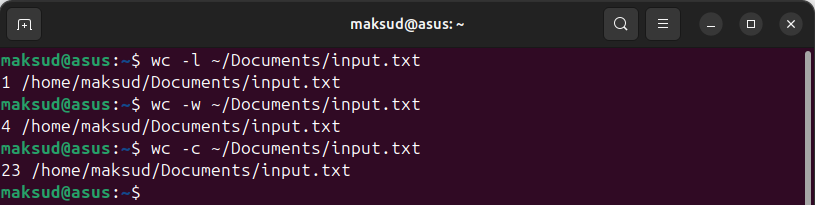
Note: “rmdir” command is used to delete empty directories. If the directory is not empty, rmdir will not delete it and will return an error.



1. Delete a directory that containing subdirectory using “**rm -r**”:

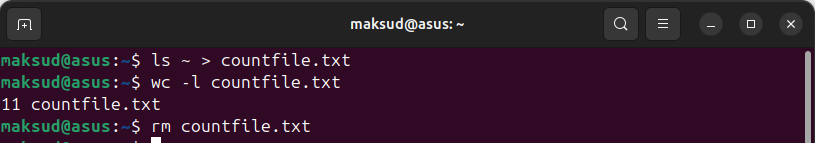


1. To count line, word, letter we can use these commad:

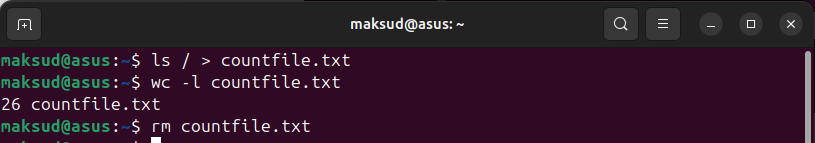


Here 1 line, 4 words, 23 letters

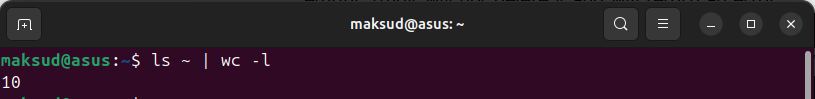
1. Counting how many files & folders are in my **home** directory using these command:



1. 20.Counting how many files & folders are in my **root** directory using these command:

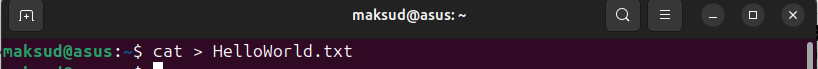


1. Pipe line:



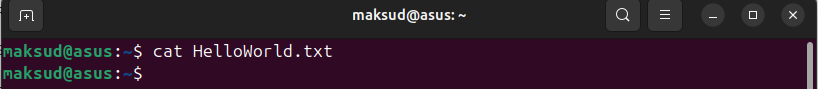
**Lab Report-03(Some files Manipulation Commands)**

1. “**$ cat > [file\_name]**” this this create, view and concatenate files.
   1. creation:



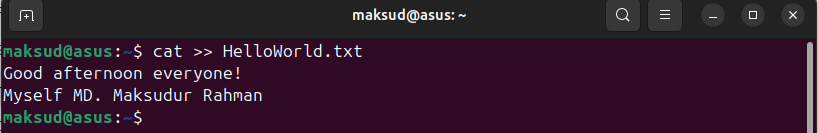
created a .txt file named “HelloWorld”

* 1. viewing:



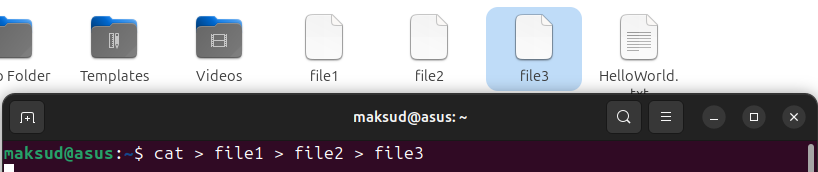
It shoows nothing becsue “HelloWorld.txt” file is empty

* 1. Add text to an existing file:

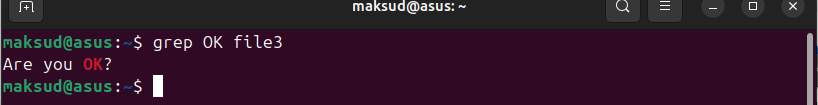


Inserted some text using “**cat >> [file\_name]**” command

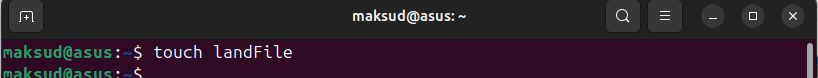
* 1. Concatenate files:



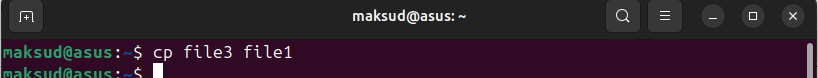
1. “**grep**”Command: Search for a pattern/word in a file



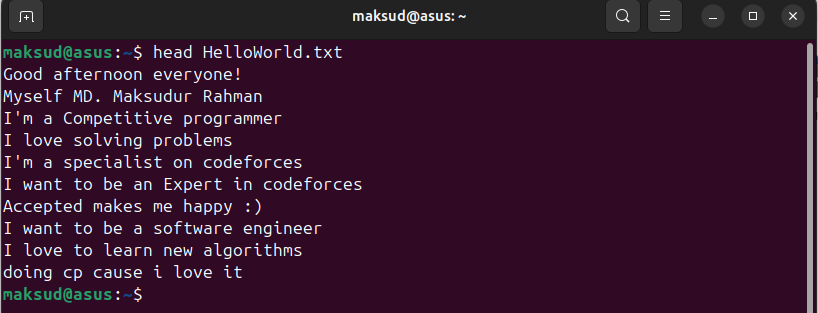
1. “**touch”**–used to create a blank file:



“**cp**” copies the file or directories

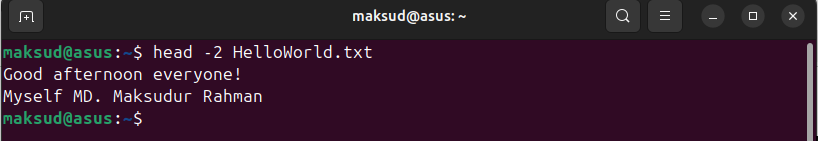


1. “**head**” displays 10 lines from the head(top)of a given file. it follows FIFO

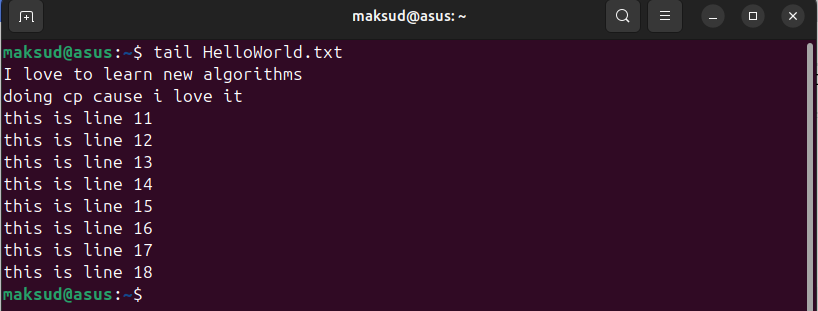


i write 11 lines of text but it display only topmost 10 lines

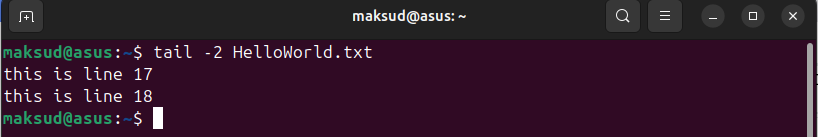
To display topmost two lines:



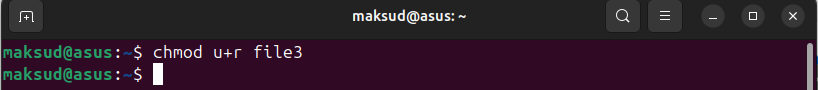
1. tail– displays last 10 lines of the file



To display last two lines



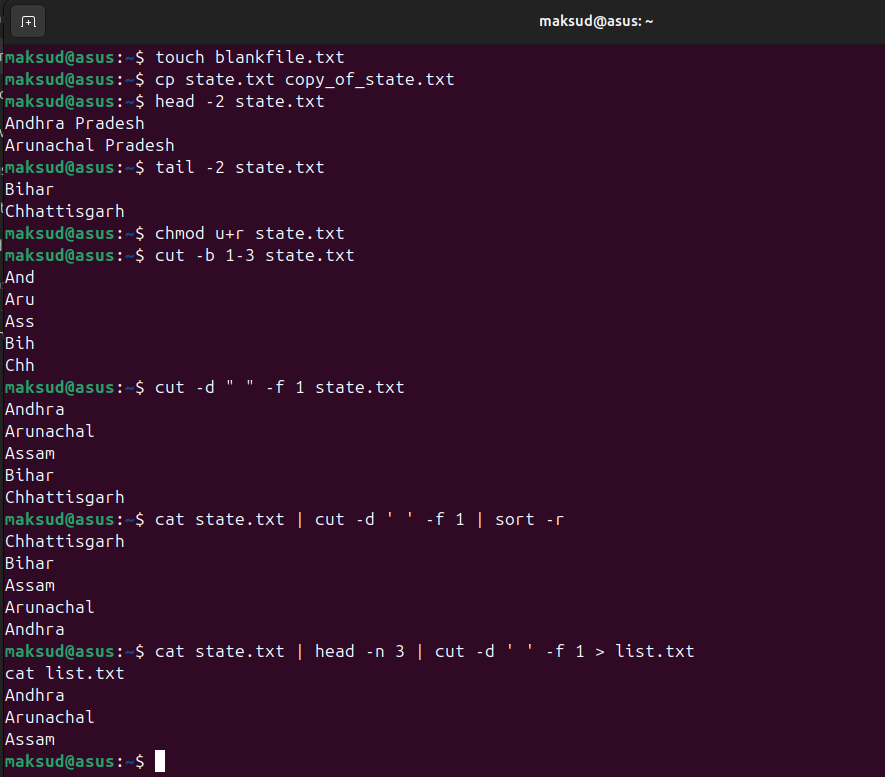
1. “**chmod**”- Change file permissions:



1. cut–it cuts or pickup a given number of character or fields of the file. It can be used to cut parts of a line by byte position, character and field. Let

us consider two files having name state.txt and capital.txt contains 5 names

of the Indian states and capitals respectively.

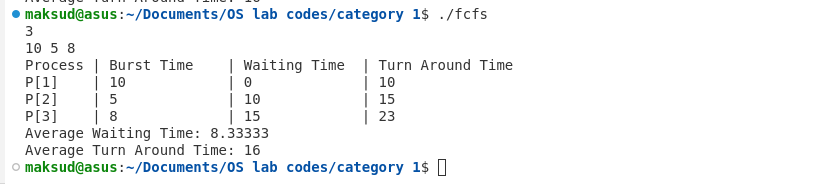


**Lab Report-04 CPU Scheduling**

**FCFS** stands for First-Come-First-Served, which is a scheduling algorithm used by the operating system to allocate CPU time to processes.This method is simple to implement but can cause longer waiting times for some processes, as demonstrated in the code mentioned below.



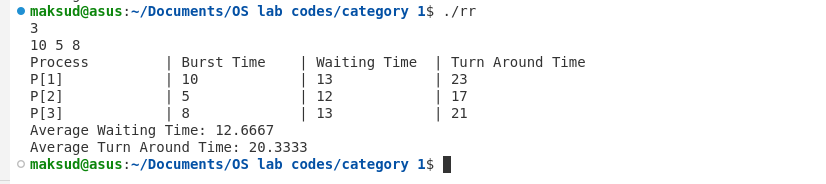
output:



**RR (Round Robin)** is a scheduling algorithm that allocates a fixed time slice (time quantum) to each process, executing them in a circular queue, providing fairness and good response time, as shown in the given code below:



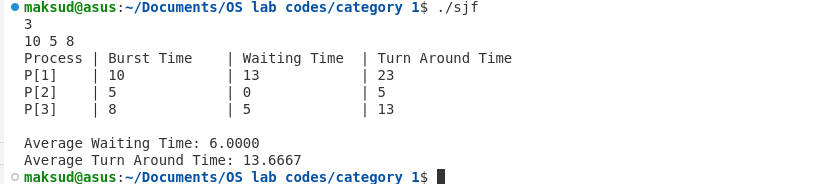
output:



**Shortest Job First (SJF)** is a CPU scheduling algorithm that selects the process with the shortest execution time to run next. It can be either preemptive or non-preemptive and aims to minimize the average waiting time, making it efficient for reducing overall process delays, as demonstrated in the code below.



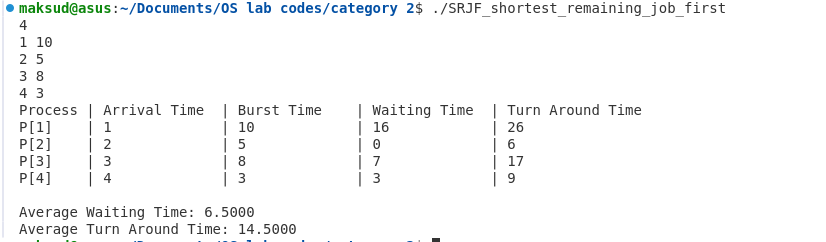
output:



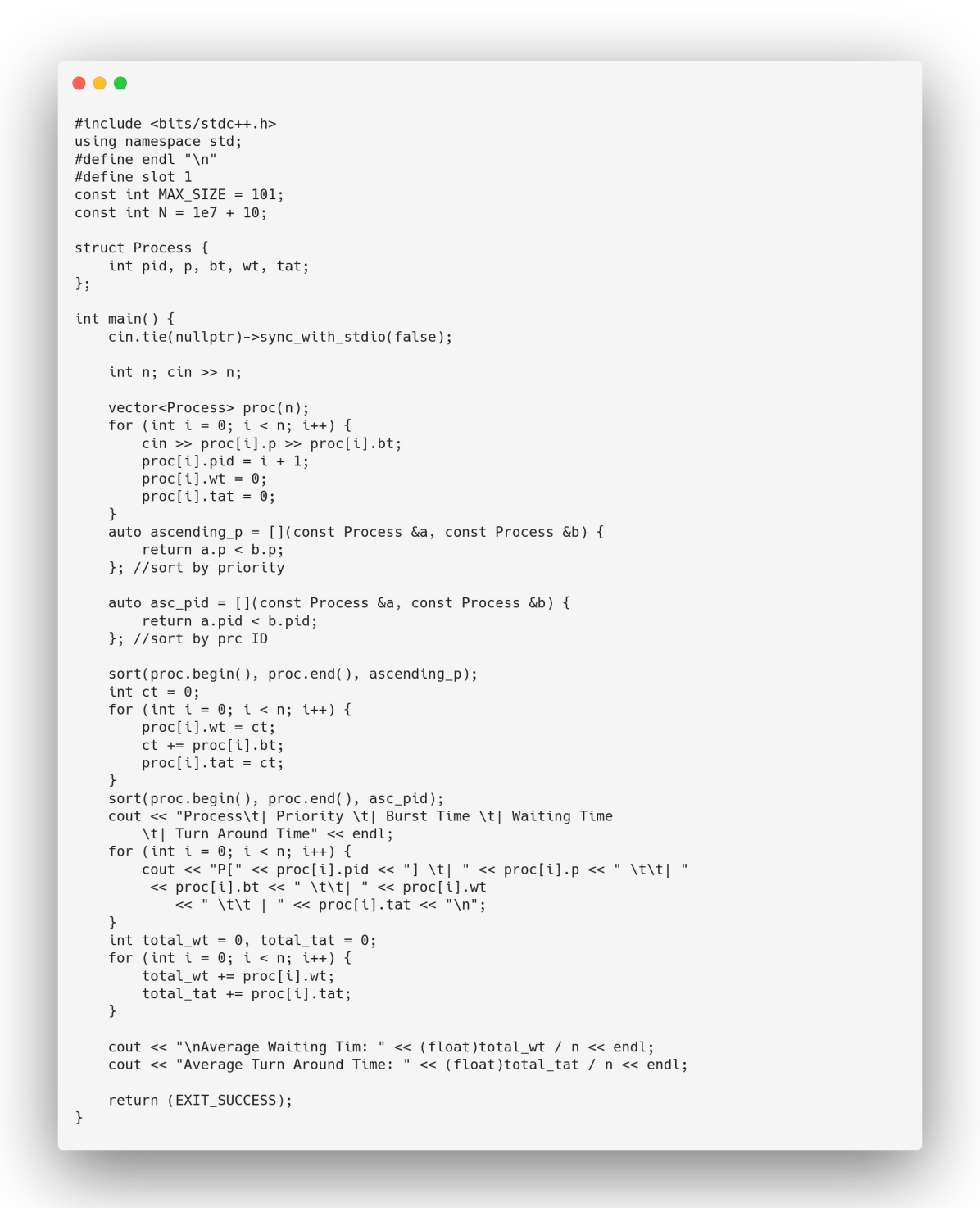
**Shortest Job Remaining First (SJRF)** also known as Preemptive SJF, is a CPU scheduling algorithm where the process with the shortest remaining time is executed next. If a new process arrives with a shorter burst time, it preempts the current one. This approach reduces overall waiting time and improves responsiveness, as shown in the code given below:



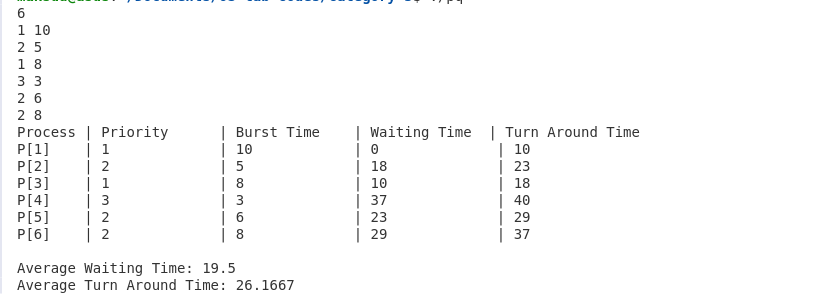
Output:



**Priority Queuing (PQ)** is a CPU scheduling algorithm where each process is assigned a priority, and the process with the highest priority is executed first. If two processes have the same priority, they are scheduled according to their arrival time. PQ can be preemptive or non-preemptive, ensuring that important tasks are handled first, as demonstrated in the code given below.



Output:



**Round Robin with Priority Queuing (RRPQ)** is a CPU scheduling algorithm that combines priority-based scheduling with the Round Robin approach. Processes are first selected based on their priority, and within the same priority level, they are executed in a Round Robin manner, with each receiving a fixed time slice. This ensures both fairness and priority handling, as shown in the code given below:

#include <bits/stdc++.h>

#define SLOT 4

using namespace std;

class Process {

public:

int pid, p, bt, wt, tat, rt;

Process() : pid(0), p(0), bt(0), wt(0), tat(0), rt(0) {}

};

bool asc\_pid(const Process& a, const Process& b) {

return a.pid < b.pid;

}

bool asc\_p(const Process& a, const Process& b) {

return a.p < b.p;

}

class Scheduler {

private:

int n;

Process proc[100];

public:

Scheduler(int n) : n(n) {}

void input() {

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

cin >> proc[i].p >> proc[i].bt;

proc[i].pid = i + 1;

proc[i].wt = 0;

proc[i].tat = 0;

proc[i].rt = proc[i].bt;

}

}

void print\_proc() {

cout << "Process | Priority | Burst Time | Waiting Time | Turn Around Time\n";

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

cout << "P[" <<proc[i].pid<< "] \t| " <<proc[i].p<< " \t| "

<<proc[i].bt<< " \t| " <<proc[i].wt<< " \t| " <<proc[i].tat<< "\n";

}

}

void execute\_proc(Process\* f\_proc[], int count, int& ct) {

if (ct)

for (int i = 0; i < count; i++)

f\_proc[i]->wt = ct;

while (true) {

bool no\_left = true;

for (int i = 0; i < count; i++)

if (f\_proc[i]->rt) { no\_left = false; break; }

if (no\_left) break;

for (int i = 0; i < count; i++) {

if (!f\_proc[i]->rt) continue;

int et = SLOT;

int rt = f\_proc[i]->rt - SLOT;

if (rt > 0) {

ct += SLOT;

f\_proc[i]->rt = rt;

} else {

ct += et + rt;

f\_proc[i]->rt = 0;

et = SLOT + rt;

f\_proc[i]->tat = ct;

}

for (int j = 0; j < count; j++)

if (j != i && f\_proc[j]->rt)

f\_proc[j]->wt += et;

}

}

}

void schedule\_execution() {

sort(proc, proc + n, asc\_p);

int i = 0, prevt = 0, count = 0;

int curr\_p = proc[0].p;

Process\* f\_proc[100];

while (i <= n) {

if (curr\_p != proc[i].p || i == n) {

execute\_proc(f\_proc, count, prevt);

if (i == n) break;

for (int j = 0; j < count; j++)

prevt = max(prevt, f\_proc[j]->tat);

curr\_p = proc[i].p;

count = 0;

}

f\_proc[count++] = &proc[i++];

}

}

void avg\_time() {

int twt = 0, ttat = 0;

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

twt += proc[i].wt;

ttat += proc[i].tat;

}

cout << "Average Waiting Time: " <<(float)twt/n <<endl;

cout << "Average Turn Around Time: " << float)ttat/n <<endl;

}

void run() {

input();

schedule\_execution();

sort(proc, proc + n, asc\_pid);

print\_proc();

cout << "\n";

avg\_time();

}

};

int main() {

int n; cin >> n;

Scheduler scheduler(n);

scheduler.run();

return 0;

}

Output:   
