***OPERATING SYSTEM***

**Practical File**

******

*Name : Rashmi*

*Clg Roll no. : 22570042*

*University Roll no. : 22033570006*

*Course : B.Sc Computer*

*Science (Hons.) IInd year*

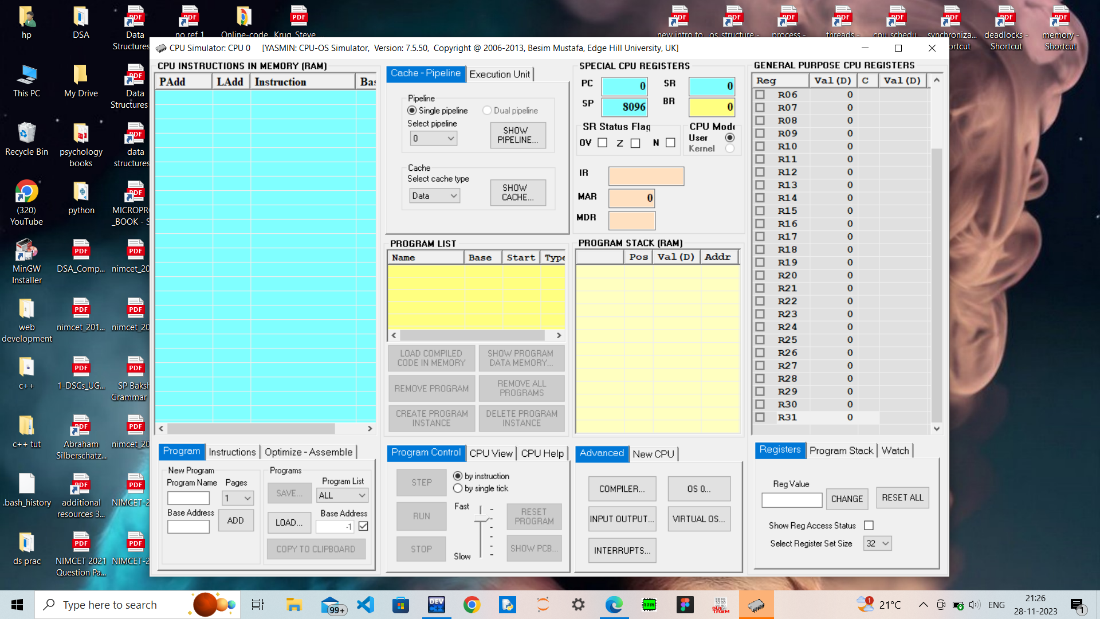
***Submitted to : Mrs Vandana Gupta***

***INDEX***

|  |  |  |
| --- | --- | --- |
| ***S.no*** | ***Practical*** | ***Page No.*** |
| ***01*** | ***Demonstration of various Operating System functions using OS Simulator.*** | ***4*** |
| ***02*** | ***Linux commands***  ***-Information maintenance***  ***- File management***  ***- Directory management*** | ***5*** |
| ***03*** | ***Linux commands***   * ***Process Control*** * ***Communication*** * ***Protection Management*** | ***13*** |
| ***04*** | ***Fork() and exce() command*** | ***18*** |
| ***05*** | ***Cpu information*** | ***23*** |
| ***06*** | ***Memory information*** | ***23*** |
| ***07*** | ***Copy file using system call*** | ***27*** |
| ***08*** | ***Fcfs algorithm*** | ***30*** |
| ***09*** | ***SJF algorithm*** | ***34*** |
| ***10*** | ***Priority based algorithm*** | ***40*** |
| ***11*** | ***Sum of n no. Using pthread*** | ***46*** |
| ***12*** | ***First-fit , best -fit , worst-fit*** | ***48*** |

***PRACTICAL 1***

***Demonstration of various Operating System functions using OS Simulator.***



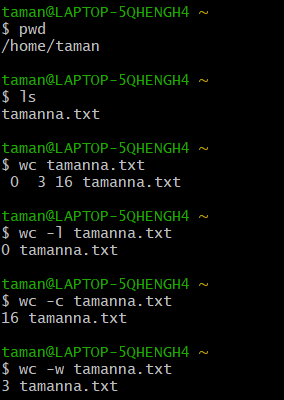
***PRACTICAL 2***

***Execute various LINUX commands for:***

1. ***Information Maintenance: wc, clear, cal, who, date, pwd***
2. ***File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk***
3. ***Directory Management: cd, mkdir, rmdir, ls***

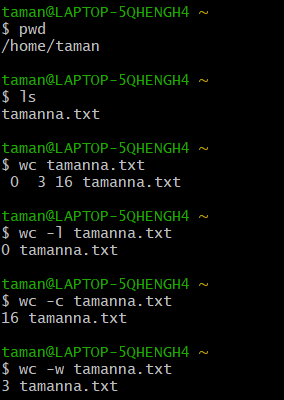
**Information Maintenance**

* `wc` (word count):

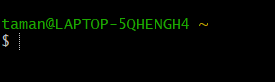


* `clear`:

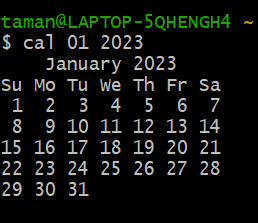
BEFORE:



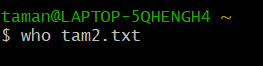
AFTER:



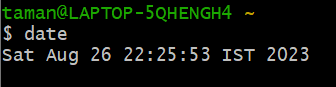
* `cal` (calendar):



* `who`:



* `date`:

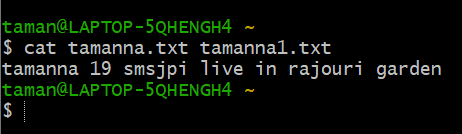


6. `pwd` (print working directory):

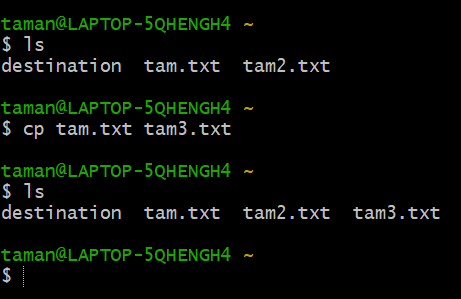


**File Management**

1. `cat`(concatenate)



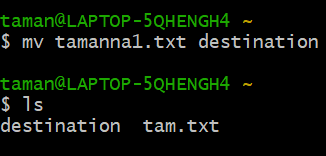
1. `cp`



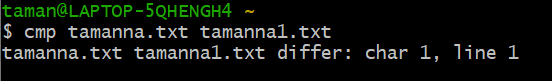
1. `rm`(remove)



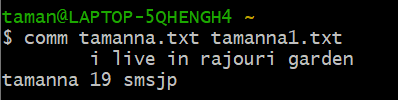
1. `mv`(move)



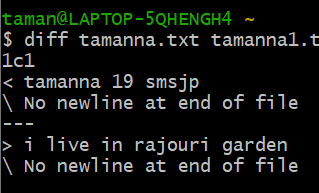
1. `cmp`



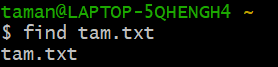
1. `comm`



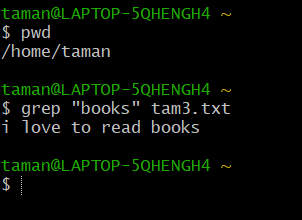
1. `diff`



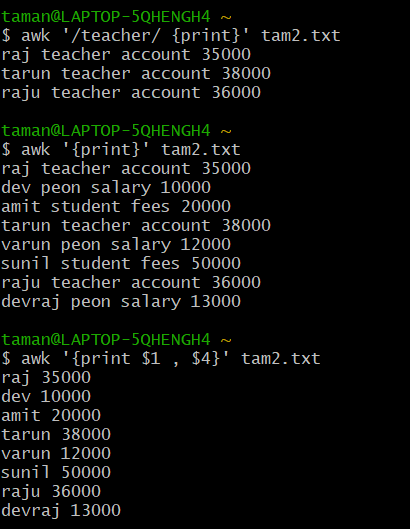
1. `find`



1. `grep`(Global Regular Expression Print

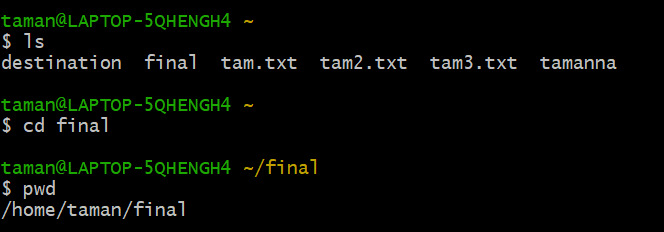


1. `awk`

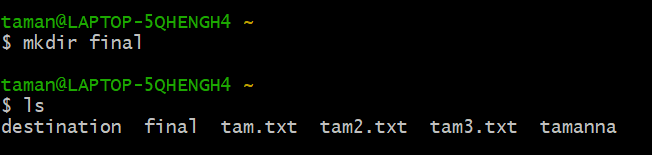


**Directory Management**

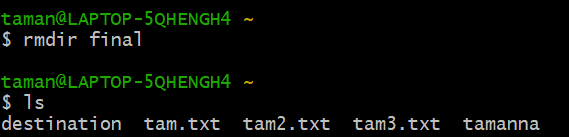
1. `cd` (change directory):



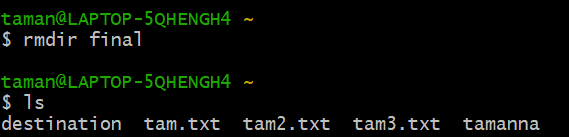
2. `mkdir` (make directory):



3. `rmdir` (remove directory):



4. `ls` (list):



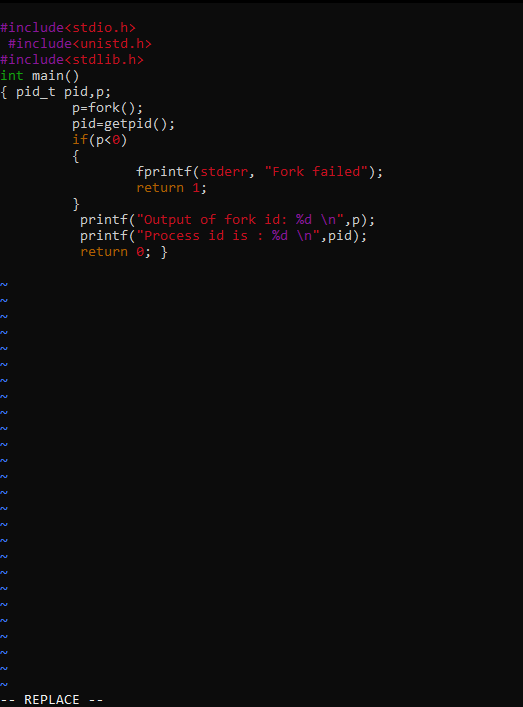
***PRACTICAL 3***

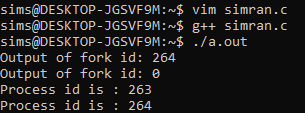
***Execute various LINUX commands for:***

1. ***Process Control: fork, getpid, ps, kill, sleep***
2. ***Communication: Input-output redirection, Pipe***
3. ***Protection Management: chmod, chown, chgrp***

### **A)Process Control**

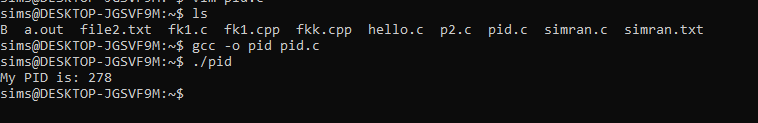
1. **Fork command**





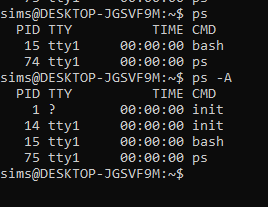
1. **Pid command**





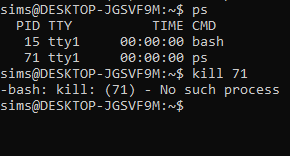
### **3) Ps command**

Output:



**4) Kill command**

Output;



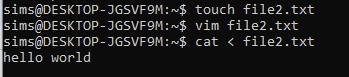
**4)Sleep command**

Output:

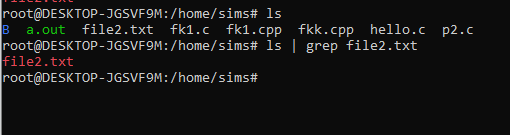


**B)Communication**

**1)Input-output redirection**



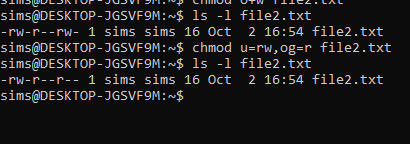
**2) Pipe**



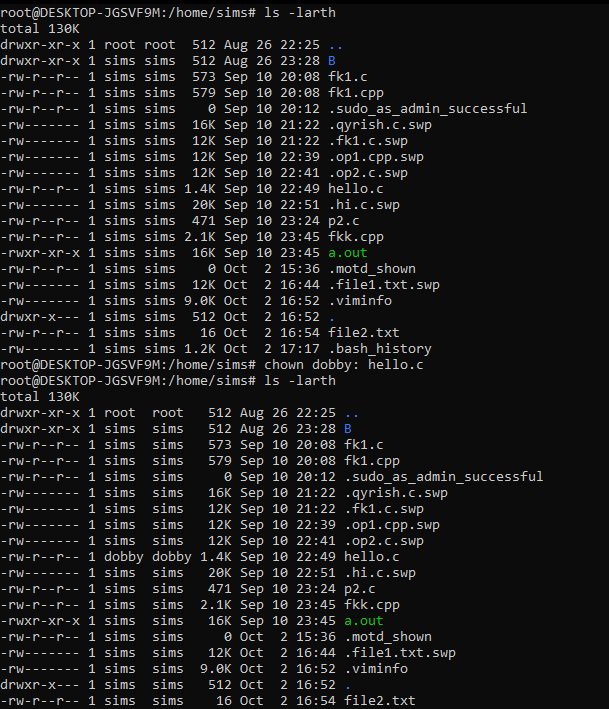
### **C) Protection Management**

**1)Chmod command**

Output:

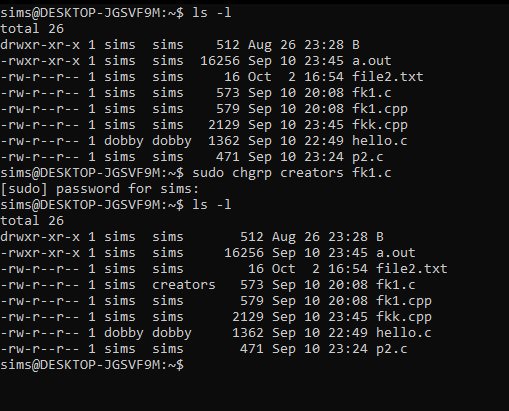


**2) chown command**

Output:

**3)chgrp command**

Output:



***PRACTICAL 4***

***Write a program (using fork() and/or exec() commands) where parent and child execute:***

***a) same program, same code.***

***b) same program, different code.***

***c) before terminating, the parent waits for the child to finish its task.***

***a)***

***#include<iostream>***

***#include<unistd.h>***

***#include<sys/types.h>***

***using namespace std;***

***int main()***

***{***

***int x;***

***int a;***

***cin>>a;***

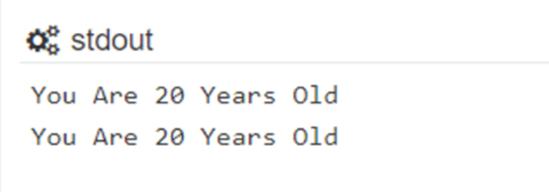
***a = 20;***

***x = fork();***

***cout<<"You Are "<<a<<" Years Old"<<endl; return 0;***

***}***

***Output:***

******

***(b).***

***#include<iostream>***

***#include<unistd.h>***

***using namespace std;***

***int main()***

***{***

***int z;***

***z=fork();***

***if(z==0)***

***{***

***cout<<"Child Process Is Running"<<endl; exit(0);***

***}***

***else***

***{***

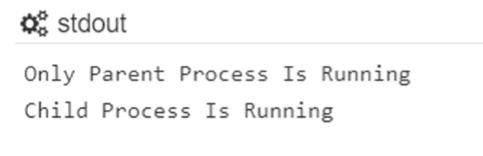
***cout<<"Only Parent Process Is Running"<<endl;***

***return 0;***

***}***

***}***

***Output:***

******

***©.***

***#include<iostream>***

***#include<wait.h>***

***#include<unistd.h>***

***using namespace std;***

***int main(){***

***int i;***

***int x;***

***x = 3;***

***i = fork();***

***if(i==0)***

***{***

***execlp("/bin/ls" , "ls",NULL); cout<<"CHILD PROCESS RUNNING\n";***

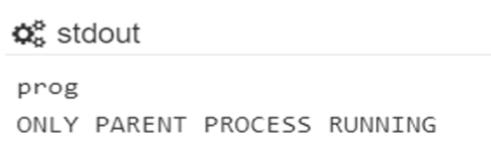
***exit(1);***

***}***

***wait(NULL);***

***cout<<"ONLY PARENT PROCESS RUNNING\n"; . return 0;***

***}***

***Output: ***

***PRACTICAL 5***

***Write a program to report behaviour of Linux kernel including kernel version, CPU type and model. (CPU information)***

***and***

***PRACTICAL 6***

***Write a program to report behaviour of Linux kernel including information on configured memory, amount of free and used memory. (memory information)***

***#include<iostream>***

***using namespace std;***

***int main() {***

***int ch;***

***cout<<"enter your choice :\n";***

***cout<<"1 for kernel version \n 2 for cpu type and model..\n 3for information of memory configured,amount of free and used memory...\n";***

***cin>>ch;***

***switch(ch){***

***case 1: cout<<"\n Kernel version is ....\n";***

***system("cat /proc/sys/kernel/osrelease");***

***break;***

***case 2: cout<<"\n cpu type & model is: \n";***

***system("awk 'NR==5{print $5}' /proc/cpuinfo");***

***break;***

***case 3: cout<<"\n amount of memory configured in system \n";***

***system("awk 'NR==4{print $0}' /proc/meminfo");***

***cout<<"\nmemory currently free in system \n";***

***system("awk 'NR==5{print $0}' /proc/meminfo");***

***cout<<"\n memory currently used in system \n";***

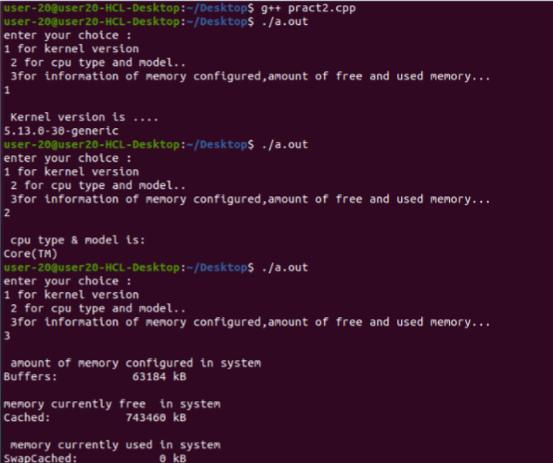
***system("awk 'NR==6{print $0}' /proc/meminfo");***

***break;***

***}***

***}***

***Output:***

******

***PRACTICAL 7***

***Write a program to copy files using system calls.***

***#include<iostream>***

***#include<fcntl.h>***

***#include<unistd.h>***

***#include<stdio.h>***

***using namespace std;***

***int main()***

***{***

***char b:***

***int fd\_one,fd\_two;***

***fd\_one-open("COPY.TXT", O\_RDONLY);***

***if(fd\_one==-1)***

***{***

***cout<<"error opening first\_filr\n";***

***close(fd\_one);***

***return 0;***

***}***

***fd\_two-open("COPIED\_FILE.TXT",O\_WRONLY|O\_CREAT.S\_IRGR PIS\_IROTH);***

***while(read(fd\_one,&b,1))***

***{***

***write(fd\_two,&b,1);***

***}***

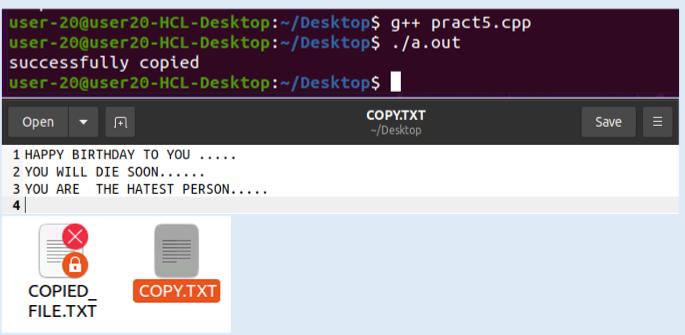
***cout<<"successfully copied"<<endl;***

***close(fd\_one);***

***close(fd\_two);***

***}***

***Output:***

******

***PRACTICAL 8***

***Write a program to implement FCFS scheduling algorithm.***

***#include <iostream>***

***using namespace std;***

***struct FCFS {***

***int at;***

***int bt;***

***int wt;***

***int tat;***

***};***

***int main() {***

***int n;***

***cout << "Enter no. of processes: ";***

***cin >> n;***

***FCFS processes[n];***

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

***cout<<"Enter arrival time of"<< i+1<<"th process :";***

***cin>>processes[i].at;***

***cout<<"Enter burst time of"<< i+1<<"th process :";***

***cin>>processes[i].bt;***

***}***

***int ct = 0;***

***double total\_wt = 0;***

***double total\_tat= 0;***

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

***if (processes[i].at > ct) {***

***ct = processes[i].at;***

***}***

***ct += processes[i].bt;***

***processes[i].tat = ct - processes[i].at;***

***processes[i].wt = processes[i].tat - processes[i].bt;***

***total\_wt += processes[i].wt;***

***total\_tat += processes[i].tat;***

***cout << "Process " << i + 1 << ": Completion Time = " << ct***

***<< ", Waiting Time = " << processes[i].wt***

***<< ", Turnaround Time = " << processes[i].tat << endl;***

***}***

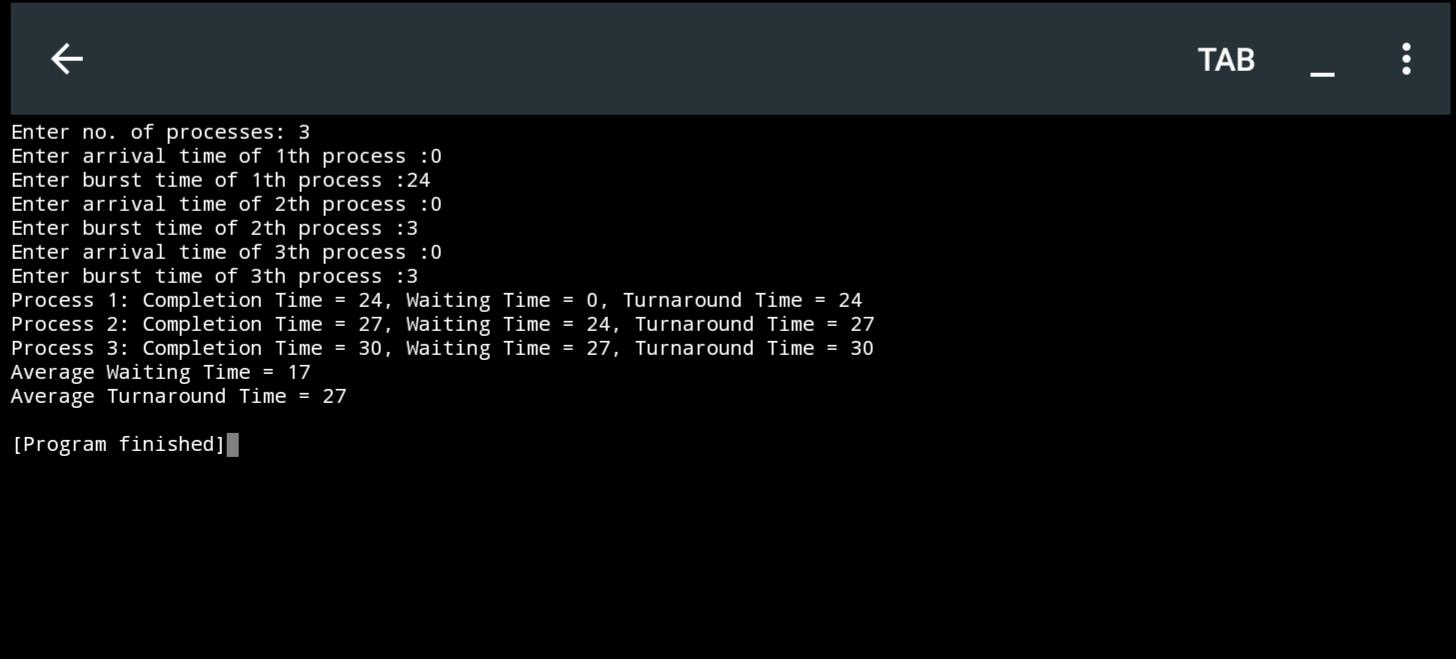
***cout << "Average Waiting Time = " << total\_wt / n << endl;***

***cout << "Average Turnaround Time = " << total\_tat / n << endl;***

***return 0;***

***}***

***Output***

******

***PRACTICAL 9***

***Write a program to implement SJF scheduling algorithm.***

***#include <iostream>***

***#include <vector>***

***#include <algorithm>***

***using namespace std;***

***struct SJF {***

***int at;***

***int bt;***

***int ct;***

***int wt;***

***int tat;***

***bool completed;***

***SJF() {***

***completed = false;***

***}***

***};***

***int main() {***

***int n;***

***cout << "Enter no. of processes: ";***

***cin >> n;***

***vector<SJF> processes(n);***

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

***cout << "Enter Arrival Time for Process " << i + 1 << ": ";***

***cin >> processes[i].at;***

***cout << "Enter Burst Time for Process " << i + 1 << ": ";***

***cin >> processes[i].bt;***

***}***

***int current\_time = 0;***

***int completed\_processes = 0;***

***while (completed\_processes < n) {***

***int shortest\_pid = -1;***

***int shortest\_bt = INT\_MAX;***

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

***if (!processes[i].completed && processes[i].at <= current\_time && processes[i].bt < shortest\_bt) {***

***shortest\_bt = processes[i].bt;***

***shortest\_pid = i;***

***}***

***}***

***if (shortest\_pid != -1) {***

***processes[shortest\_pid].ct = current\_time + processes[shortest\_pid].bt;***

***processes[shortest\_pid].tat = processes[shortest\_pid].ct - processes[shortest\_pid].at;***

***processes[shortest\_pid].wt = processes[shortest\_pid].tat - processes[shortest\_pid].bt;***

***processes[shortest\_pid].completed = true;***

***current\_time = processes[shortest\_pid].ct;***

***completed\_processes++;***

***} else {***

***current\_time++;***

***}***

***}***

***float total\_wt = 0;***

***float total\_tat = 0;***

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

***total\_wt += processes[i].wt;***

***total\_tat += processes[i].tat;***

***}***

***float average\_wt = total\_wt / n;***

***float average\_tat = total\_tat / n;***

***cout << "\nProcess\tCompletion Time\tWaiting Time\tTurnaround Time\n";***

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

***cout << "P" << i + 1 << "\t\t" << processes[i].ct << "\t\t"***

***<< processes[i].wt << "\t\t" << processes[i].tat << endl;***

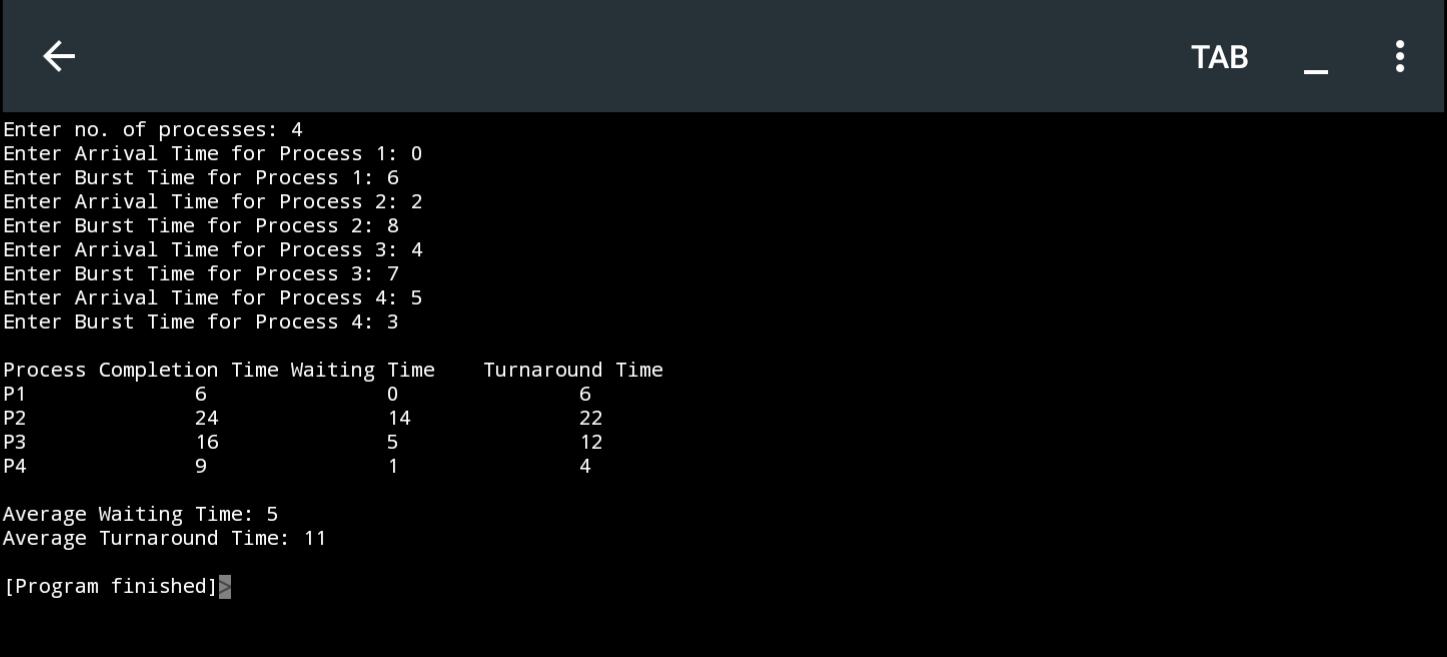
***}***

***cout << "\nAverage Waiting Time: " << average\_wt << endl;***

***cout << "Average Turnaround Time: " << average\_tat << endl;***

***return 0;***

***}***

***Output***

***PRACTICAL 10***

***Write a program to implement non-preemptive priority based scheduling algorithm.***

***#include <iostream>***

***#include <algorithm>***

***using namespace std;***

***const int MAX\_PROCESSES = 100;***

***struct Process {***

***int pid;***

***int bt;***

***int priority;***

***};***

***bool compare(const Process& p1, const Process& p2) {***

***return p1.priority < p2.priority;***

***}***

***void priorityScheduling(Process processes[], int numProcesses) {***

***// Sort processes based on priority***

***sort(processes, processes + numProcesses, compare);***

***int totalTime = 0;***

***for (int i = 0; i < numProcesses; ++i) {***

***totalTime += processes[i].bt;***

***}***

***float average\_wt = 0.0;***

***float average\_tat = 0.0;***

***int wt[MAX\_PROCESSES] = {0};***

***int tat[MAX\_PROCESSES] = {0};***

***wt[0] = 0;***

***tat[0] = processes[0].bt;***

***for (int i = 1; i < numProcesses; ++i) {***

***wt[i] = tat[i - 1];***

***tat[i] = wt[i] + processes[i].bt;***

***}***

***for (int i = 0; i < numProcesses; ++i) {***

***average\_wt += wt[i];***

***average\_tat += tat[i];***

***}***

***average\_wt /= numProcesses;***

***average\_tat /= numProcesses;***

***cout << "Process\tBurst Time\tPriority\tWait Time\tTurnaround Time\n";***

***for (int i = 0; i < numProcesses; ++i) {***

***cout << processes[i].pid << "\t" << processes[i].bt<< "\t\t"***

***<< processes[i].priority << "\t\t" << wt[i] << "\t\t" << tat[i] << "\n";***

***}***

***cout << "\nAverage Wait Time: " << average\_wt << "\n";***

***cout << "Average Turnaround Time: " << average\_tat << "\n";***

***}***

***int main() {***

***using namespace std;***

***int numProcesses;***

***cout << "Enter the number of processes: ";***

***cin >> numProcesses;***

***Process processes[MAX\_PROCESSES];***

***for (int i = 0; i < numProcesses; ++i) {***

***processes[i].pid = i + 1;***

***cout << "Enter burst time for Process " << i + 1 << ": ";***

***cin >> processes[i].bt;***

***cout << "Enter priority for Process " << i + 1 << ": ";***

***cin >> processes[i].priority;***

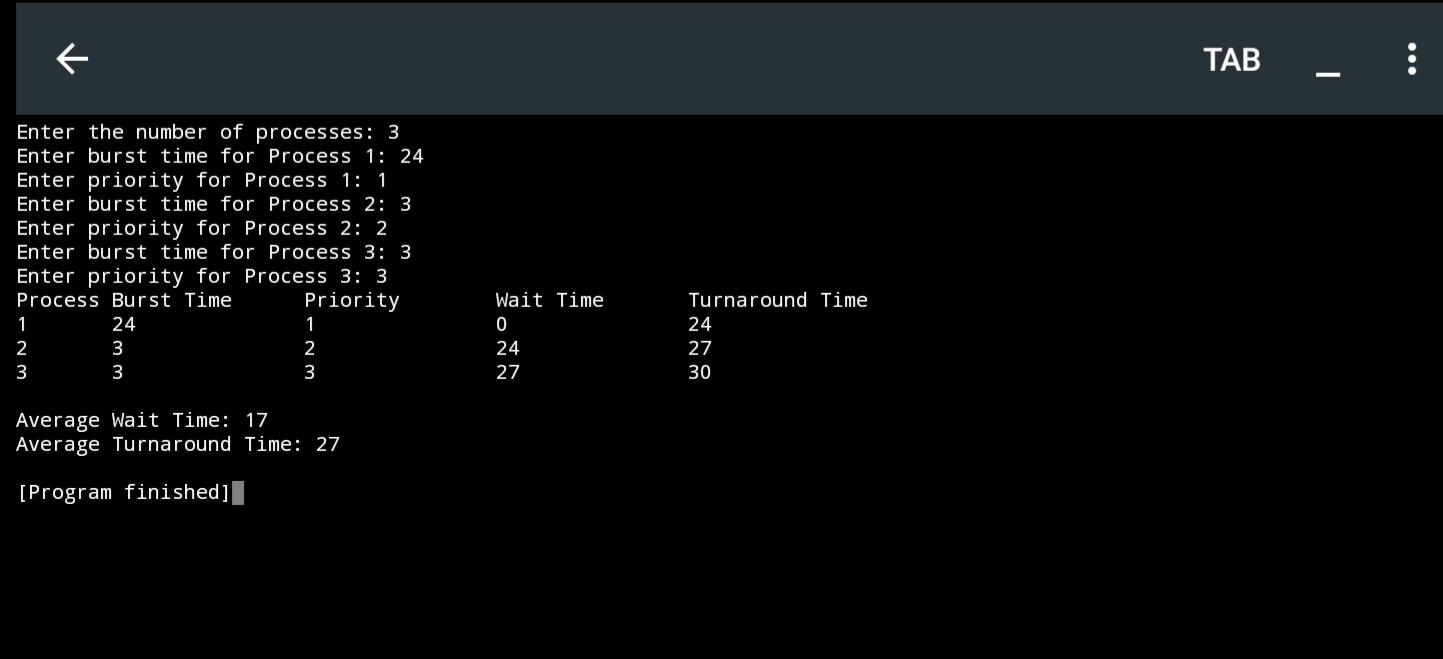
***}***

***priorityScheduling(processes, numProcesses);***

***return 0;***

***}***

***Output***

******

***PRACTICAL 11***

***Write a program to calculate sum of n numbers using pthread.***

***#include<iostream>***

***#include<pthread.h>***

***using namespace std;***

***int global[2];***

***void \*sum\_thread(void \*arg)***

***{***

***int \*args\_array;***

***args\_array=(int\*)arg;***

***int n1,n2,sum;***

***n1=global[0];***

***n2=global[1];***

***sum=n1+n2;***

***cout<<"\n sum= "<<sum;***

***return NULL;***

***}***

***int main() {***

***cout<<"\n First Number: ";***

***cin>>global[0];***

***cout<<"\n second Number: ";***

***cin>>global[1];***

***pthread\_t tid\_sum;***

***pthread\_create(&tid\_sum,NULL,sum\_thread,(void\*)&global);***

***pthread\_join(tid\_sum,NULL);***

***return 0;***

***}***

***Output:***

******

***PRACTICAL 12***

***Write a program to implement first-fit, best-fit and worst-fit allocation strategies.***

***#include <iostream>***

***#include <vector>***

***#include <algorithm>***

***struct MemoryBlock {***

***int blockID;***

***int size;***

***bool allocated;***

***};***

***void displayMemory(const std::vector<MemoryBlock>& memory) {***

***std::cout << "Memory Blocks:\n";***

***std::cout << "BlockID\tSize\tAllocated\n";***

***for (const MemoryBlock& block : memory) {***

***std::cout << block.blockID << "\t" << block.size << "\t" << (block.allocated ? "Yes" : "No") << "\n";***

***}***

***std::cout << "\n";***

***}***

***void firstFit(std::vector<MemoryBlock>& memory, int processSize) {***

***for (MemoryBlock& block : memory) {***

***if (!block.allocated && block.size >= processSize) {***

***block.allocated = true;***

***std::cout << "First-Fit: Allocated process of size " << processSize << " to block " << block.blockID << "\n";***

***return;***

***}***

***}***

***std::cout << "First-Fit: Unable to allocate process of size " << processSize << "\n";***

***}***

***void bestFit(std::vector<MemoryBlock>& memory, int processSize) {***

***auto bestFitBlock = std::find\_if(memory.begin(), memory.end(), [processSize](const MemoryBlock& block) {***

***return !block.allocated && block.size >= processSize;***

***});***

***if (bestFitBlock != memory.end()) {***

***bestFitBlock->allocated = true;***

***std::cout << "Best-Fit: Allocated process of size " << processSize << " to block " << bestFitBlock->blockID << "\n";***

***} else {***

***std::cout << "Best-Fit: Unable to allocate process of size " << processSize << "\n";***

***}***

***}***

***void worstFit(std::vector<MemoryBlock>& memory, int processSize) {***

***auto worstFitBlock = std::max\_element(memory.begin(), memory.end(), [](const MemoryBlock& block1, const MemoryBlock& block2) {***

***return block1.size < block2.size;***

***});***

***if (worstFitBlock != memory.end() && worstFitBlock->size >= processSize) {***

***worstFitBlock->allocated = true;***

***std::cout << "Worst-Fit: Allocated process of size " << processSize << " to block " << worstFitBlock->blockID << "\n";***

***} else {***

***std::cout << "Worst-Fit: Unable to allocate process of size " << processSize << "\n";***

***}***

***}***

***int main() {***

***using namespace std;***

***int numBlocks;***

***cout << "Enter the number of memory blocks: ";***

***cin >> numBlocks;***

***vector<MemoryBlock> memory(numBlocks);***

***for (int i = 0; i < numBlocks; ++i) {***

***memory[i].blockID = i + 1;***

***cout << "Enter size of Memory Block " << i + 1 << ": ";***

***cin >> memory[i].size;***

***memory[i].allocated = false;***

***}***

***displayMemory(memory);***

***int processSize;***

***cout << "Enter the size of the process: ";***

***cin >> processSize;***

***firstFit(memory, processSize);***

***displayMemory(memory);***

***bestFit(memory, processSize);***

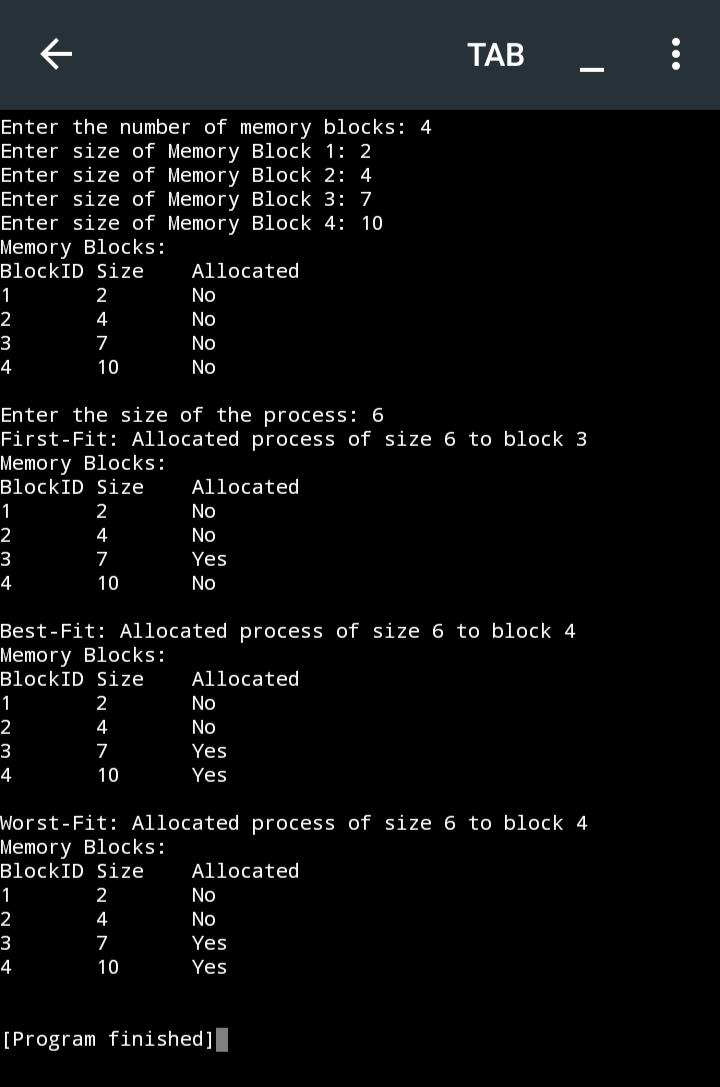
***displayMemory(memory);***

***worstFit(memory, processSize);***

***displayMemory(memory);***

***return 0;***

***}***

******