Operating System Practical

UNIVERSITY OF DELHI

Ramanujan College

DSC 08: Operating System

Semester-3

(2024-25)

Submitted By:

~Ratnesh Kumar

20231432(23020570034)

Submitted To:

~Ms. Sheetal Singh

(Assistant Professor)

Department of Computer Science

Acknowledgement

I would like to take this opportunity to acknowledge everyone who has helped us in every stage of this project.

I am deeply indebted to my Operating System Professor, Ms. Sheetal Singh for his guidance and suggestions in completing this project. The completion of this project was possible under his guidance and support.

I am also very thankful to my parents and my friends who have boosted me up morally with their continuous support.

At last but not least, I am very thankful to God almighty for showering his blessings upon me.

Index

1. Execute various LINUX commands for:

i. Information Maintenance: wc, clear, cal, who, date, pwd

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ wc demo.txt
  7 129 702 demo.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ cal
   November 2024
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ who
ratneshkumar pts/1        2024-11-24 12:49
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ date
Sun Nov 24 13:26:19 UTC 2024
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ pwd
/mnt/c/Users/user/Documents/OS_Practical
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ clear
```

ii. File Management: cat, cp, rm, mv, cmp, comm, diff, find, grep, awk

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ cat demo.txt
A hare and tortoise lived in a distant jungle. But the hare always mocked the tortoise.
"You are so slow, do you ever make it on time?"
Offended by the rude tone, the tortoise replied in a humble tone.
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ cp demo.txt copy.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ ls
copy.txt  demo.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ mv copy.txt duplicate.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ comm demo.txt duplicate.txt
                A hare and tortoise lived in a distant jungle. But the hare always mocked the tortoise.
                "You are so slow, do you ever make it on time?"
                Offended by the rude tone, the tortoise replied in a humble tone.
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ diff demo.txt duplicate.txt
3a4
> duplicate by ratnesh
\ No newline at end of file
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ cmp demo.txt duplicate.txt
cmp: EOF on demo.txt after byte 209, line 3
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ find ./ -name demo.txt./demo.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ grep -i ratnesh duplicate.txt
duplicate by ratnesh
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ awk '{print $2, $3}' demo.txt
hare and
are so
by the
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ rm duplicate.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ ls
demo.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$
```

iii. Directory Management: cd, mkdir, rmdir, ls

```
ratneshkumar@RATNESH-KUMAR:~$ mkdir OS_Practical
ratneshkumar@RATNESH-KUMAR:~$ ls
OS_Practical  snap
ratneshkumar@RATNESH-KUMAR:~$ cd OS_Practical/
ratneshkumar@RATNESH-KUMAR:~/OS_Practical$ cd ../
ratneshkumar@RATNESH-KUMAR:~$ rmdir OS_Practical/
ratneshkumar@RATNESH-KUMAR:~$ ls
snap
ratneshkumar@RATNESH-KUMAR:~$
```

2. Execute various LINUX commands for:

i. Process Control: fork, getpid, ps, kill, sleep

```cpp
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
int main() {
    std::cout << "Process Control Demonstration:\n";
    pid_t pid = fork();
    if (pid == 0) {
        std::cout << "Child process running. PID: " << getpid() << std::endl;
        sleep(2);
        std::cout << "Child process finished.\n";
        exit(0);
    } else if (pid > 0) {
        std::cout << "Parent process running. PID: " << getpid() << std::endl;
        std::cout << "Waiting for child process to finish...\n";
        wait(nullptr);
    } else {
        std::cerr << "Fork failed.\n";
        return 1;}
    std::cout << "Listing current processes using 'ps':\n";
    system("ps");
    std::cout << "Demonstrating kill:"<<getpid()<<"\n";
    kill(getpid(), SIGTERM);}
```

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Process Control'
Process Control Demonstration:
Parent process running. PID: 1344
Waiting for child process to finish...
Child process running. PID: 1345
Child process finished.
Listing current processes using 'ps':
    PID TTY          TIME CMD
    394 pts/0    00:00:00 bash
   1344 pts/0    00:00:00 Process Control
   1346 pts/0    00:00:00 sh
   1347 pts/0    00:00:00 ps
Demonstrating kill:1344
Terminated
```

ii. Communication: Input-output redirection, Pipe

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ ls
demo.txt  output.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ echo "Hello Ratnesh" > output.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ cat output.txt
Hello Ratnesh
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ wc -l < demo.txt
3
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$ ls -l |grep ".txt"
-rwxrwxrwx 1 ratneshkumar ratneshkumar 209 Nov 24 13:38 demo.txt
-rwxrwxrwx 1 ratneshkumar ratneshkumar  14 Nov 24 16:51 output.txt
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical$
```

iii. Protection Management: chmod, chown, chgrp

```
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# ls
demo.txt  output.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# chmod 755 demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# ls -l demo.txt
-rwxrwxrwx 1 ratneshkumar ratneshkumar 209 Nov 24 13:38 demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# chown root:root demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# ls -l demo.txt
-rwxrwxrwx 1 ratneshkumar ratneshkumar 209 Nov 24 13:38 demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# chgrp staff demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical# ls -l demo.txt
-rwxrwxrwx 1 ratneshkumar ratneshkumar 209 Nov 24 13:38 demo.txt
root@RATNESH-KUMAR:/mnt/c/Users/user/Documents/OS_Practical#
```

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

i. same program, same code.

ii. same program, different code.

iii. before terminating, the parent waits for the child to finish its task.

Code:-

```
#include <iostream>
#include <unistd.h>
#include <sys/wait.h>
using namespace std;
void execute_same_program_same_code() {
   cout << "Executing the same program with same code.\n";}
void execute_same_program_different_code() {
   cout << "Parent is executing its task.\n";
   execlp("pwd", "pwd",nullptr);
   perror("exec failed");}
int main() {
   pid_t pid = fork();
   if (pid == -1) {
      cerr << "Fork failed!\n";
      return 1;}
   if (pid == 0) {
      cout << "Child process (PID: " << getpid() << ") started.\n";
      execute_same_program_same_code();
      execute_same_program_different_code();
   } else {
      cout << "Parent process (PID: " << getpid() << ") started.\n";
      execute_same_program_same_code();
      cout << "Parent is waiting for child to finish...\n";
      wait(NULL);
      cout << "Child has finished. Parent is now terminating.\n";}
   return 0;}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./Use\ fork\ \(\)\ and\ exec\
\(\)\ commands
Parent process (PID: 1395) started.
Executing the same program with same code.
Parent is waiting for child to finish...
Child process (PID: 1396) started.
Executing the same program with same code.
Parent is executing its task.
/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++
Child has finished. Parent is now terminating.
```

4. Write a program to report behaviour of Linux kernel including kernel version, CPU type and CPU information.

Code:-

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main() {
   printf("Kernel Version:\n");
   system("uname -r");
   printf("\nCPU Type:\n");
   system("uname -m");
   printf("\nDetailed CPU Information:\n");
   system("lscpu");
   return 0;}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Report behaviour of Linux k
ernel including kernel version, CPU type and CPU  information'
Kernel Version:
5.15.167.4-microsoft-standard-WSL2

CPU Type:
x86_64

Detailed CPU Information:
Architecture:            x86_64
  CPU op-mode(s):        32-bit, 64-bit
  Address sizes:         39 bits physical, 48 bits virtual
  Byte Order:            Little Endian
CPU(s):                  4
  On-line CPU(s) list:   0-3
Vendor ID:               GenuineIntel
  Model name:            Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz
    CPU family:          6
    Model:               78
    Thread(s) per core:  2
    Core(s) per socket:  2
    Socket(s):           1
    Stepping:            3
    BogoMIPS:            4799.99
```

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

Code:-

```
#include <iostream>
#include <fstream>
#include <string>
using namespace std;
void get_memory_info() {
   ifstream meminfo("/proc/meminfo");
   string line;
   if (!meminfo.is_open()) {
      cerr << "Failed to open /proc/meminfo" << endl;
      exit(1);}
   cout << "\nMemory Information:\n";
   while (getline(meminfo, line)) {
      if (line.find("MemTotal") != string::npos) {
         cout << line << endl;}
      if (line.find("MemFree") != string::npos) {
         cout << line << endl; }
      if (line.find("MemAvailable") != string::npos) {
         cout << line << endl;}
      if (line.find("Buffers") != string::npos) {
         cout << line << endl; }
      if (line.find("Cached") != string::npos) {
         cout << line << endl;}
      if (line.find("SwapTotal") != string::npos) {
         cout << line << endl; }
      if (line.find("SwapFree") != string::npos) {
         cout << line << endl;}}
   meminfo.close();}
int main() {
   cout << "Linux Kernel Memory Information:\n";
   get_memory_info();
   return 0;}
```

Output:-

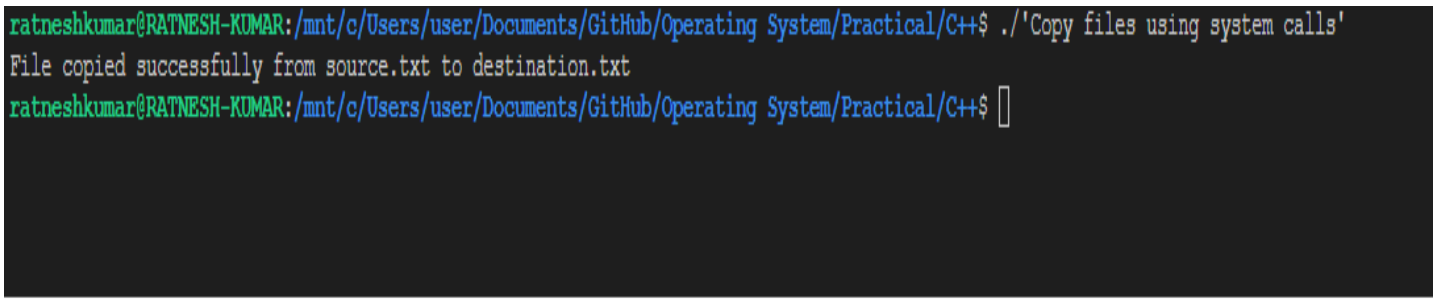
```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Report behaviour of Linux k
ernel including information on configured memory, amount of  free and used memory'
Linux Kernel Memory Information:

Memory Information:
MemTotal:        3953528 kB
MemFree:         3408748 kB
MemAvailable:    3373940 kB
Buffers:            1072 kB
Cached:           127744 kB
SwapCached:            0 kB
SwapTotal:       1048576 kB
SwapFree:        1048576 kB
```

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

Code:-

```
#include <iostream>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
using namespace std;
void copy_file(const char* source, const char* destination) {
   int source_fd = open(source, O_RDONLY);
   if (source_fd == -1) {
      perror("Failed to open source file");
      exit(1);}
   int dest_fd = open(destination, O_WRONLY | O_CREAT | O_TRUNC, S_IRUSR | S_IWUSR);
   if (dest_fd == -1) {
      perror("Failed to open destination file");
      close(source_fd);
      exit(1);}
   char buffer[1024];
   ssize_t bytes_read, bytes_written;
   while ((bytes_read = read(source_fd, buffer, sizeof(buffer))) > 0) {
      bytes_written = write(dest_fd, buffer, bytes_read);
      if (bytes_written != bytes_read) {
         perror("Error writing to destination file");
         close(source_fd);
         close(dest_fd);
         exit(1);}}
   if (bytes_read == -1) {
      perror("Error reading from source file");}
   close(source_fd);
   close(dest_fd);
   cout << "File copied successfully from " << source << " to " << destination << endl;}
int main() {
   const char* source = "source.txt";
   const char* destination = "destination.txt";
   copy_file(source, destination);
   return 0;}
```

Output:-



7. Write a program to implement FCFS scheduling algorithm.

Code:-

```
#include <iostream>
void First_Come_First_Serve(std::string name_of_process[], int burst_time[],int number_of_processes){
   double waiting_time = 0;
   double turnaround_time = 0;
   double total_waiting_time = 0;
   double total_turnaround_time = 0;
   std::cout<<"Name Of Process\tBurst Time\tWaiting Time\tTurnAround Time"<<std::endl;
   for (int i=0;i<number_of_processes;i++){
      turnaround_time += burst_time[i];
      total_turnaround_time+=turnaround_time;
std::cout<<name_of_process[i]<<"\t\t"<<burst_time[i]<<"\t\t"<<waiting_time<<"\t\t"<<turnaround_time<<std:
:endl;
     total_waiting_time+=waiting_time;
     waiting_time += burst_time[i];}
   std::cout<<"\nAverage Waiting Time is:: "<<total_waiting_time/number_of_processes<<"\nAverage
TurnAround Time is::"<<total_turnaround_time/number_of_processes<<std::endl;}
int main(){
   int number_of_processes;
   std::cout<<"Enter the number of processes: ";
   std::cin>>number_of_processes;
   std::string name_of_process[number_of_processes];
   int burst_time[number_of_processes];
   for (int i=0;i<number_of_processes;i++){
      std::cout<<"Enter Name of Process:: ";
      std::cin>>name_of_process[i];
      std::cout<<"Enter Burst Time of Process "<<name_of_process[i]<<":: ";
      std::cin>>burst_time[i];}
   First_Come_First_Serve(name_of_process,burst_time,number_of_processes);}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'First Come First Serve'
Enter the number of processes: 4
Enter Name of Process:: p1
Enter Burst Time of Process p1:: 5
Enter Name of Process:: p2
Enter Burst Time of Process p2:: 3
Enter Name of Process:: p3
Enter Burst Time of Process p3:: 8
Enter Name of Process:: p4
Enter Burst Time of Process p4:: 10
Name Of Process Burst Time      Waiting Time    TurnAround Time
p1              5               0               5
p2              3               5               8
p3              8               8               16
p4              10              16              26

Average Waiting Time is:: 7.25
Average TurnAround Time is::13.75
```

8. Write a program to implement SJF scheduling algorithm.

Code:-

```
#include <iostream>
void First_Come_First_Serve(std::string name_of_process[], int burst_time[],int number_of_processes){
   double waiting_time = 0;
   double turnaround_time = 0;
   double total_waiting_time = 0;
   double total_turnaround_time = 0;
   std::cout<<"Name Of Process\tBurst Time\tWaiting Time\tTurnAround Time"<<std::endl;
   for (int i=0;i<number_of_processes;i++){
      turnaround_time += burst_time[i];
      total_turnaround_time+=turnaround_time;
std::cout<<name_of_process[i]<<"\t\t"<<burst_time[i]<<"\t\t"<<waiting_time<<"\t\t"<<turnaround_time<<std::endl;
     total_waiting_time+=waiting_time;
     waiting_time += burst_time[i];}
   std::cout<<"\nAverage Waiting Time is:: "<<total_waiting_time/number_of_processes<<"\nAverage TurnAround Time
is::"<<total_turnaround_time/number_of_processes<<std::endl;}
void Sort(std::string name_of_process[], int burst_time[],int number_of_processes){
   for(int i=0;i<number_of_processes;i++){
      for(int j=0;j<number_of_processes;j++){
         if (burst_time[i] < burst_time[j]){
            int temp_burst = burst_time[i];
            burst_time[i] = burst_time[j];
            burst_time[j] = temp_burst;
            std::string temp_name_of_process = name_of_process[i];
            name_of_process[i] = name_of_process[j];
            name_of_process[j]= temp_name_of_process;}}}
   First_Come_First_Serve(name_of_process,burst_time,number_of_processes);}
int main(){
   int number_of_processes;
   std::cout<<"Enter the number of processes: ";
   std::cin>>number_of_processes;
   std::string name_of_process[number_of_processes];
   int burst_time[number_of_processes];
   for (int i=0;i<number_of_processes;i++){
      std::cout<<"Enter Name of Process:: ";
      std::cin>>name_of_process[i];
      std::cout<<"Enter Burst Time of Process "<<name_of_process[i]<<":: ";
      std::cin>>burst_time[i];}
   Sort(name_of_process,burst_time,number_of_processes);}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Shortest Job First'
Enter the number of processes: 3
Enter Name of Process:: p1
Enter Burst Time of Process p1:: 9
Enter Name of Process:: p2
Enter Burst Time of Process p2:: 5
Enter Name of Process:: p3
Enter Burst Time of Process p3:: 20
Name Of Process Burst Time      Waiting Time    TurnAround Time
p2              5               0               5
p1              9               5               14
p3              20              14              34

Average Waiting Time is:: 6.33333
Average TurnAround Time is::17.6667
```

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

Code:-

```
#include <iostream>
void First_Come_First_Serve(std::string name_of_process[], int burst_time[],int priority[],int
number_of_processes){
   double waiting_time = 0;
   double turnaround_time = 0;
   double total_waiting_time = 0;
   double total_turnaround_time = 0;
   std::cout<<"Name Of Process\tBurst Time\tPriority\tWaiting Time\tTurnAround Time"<<std::endl;
   for (int i=0;i<number_of_processes;i++){
      turnaround_time += burst_time[i];
      total_turnaround_time+=turnaround_time;
std::cout<<name_of_process[i]<<"\t\t"<<burst_time[i]<<"\t\t"<<priority[i]<<"\t\t"<<waiting_time<<"\t\t"<<tur
naround_time<<std::endl;
      total_waiting_time+=waiting_time;
      waiting_time += burst_time[i];}
   std::cout<<"\nAverage Waiting Time is:: "<<total_waiting_time/number_of_processes<<"\nAverage
TurnAround Time is::"<<total_turnaround_time/number_of_processes<<std::endl;}
void Sort(std::string name_of_process[], int burst_time[],int priority[],int number_of_processes){
   for(int i=0;i<number_of_processes;i++){
      for(int j=0;j<number_of_processes;j++){
         if (priority[i] < priority[j]){
            int temp_burst = burst_time[i];
            burst_time[i] = burst_time[j];
            burst_time[j] = temp_burst;
            int temp_priority = priority[i];
            priority[i] = priority[j];
            priority[j] = temp_priority;
            std::string temp_name_of_process = name_of_process[i];
            name_of_process[i] = name_of_process[j];
            name_of_process[j]= temp_name_of_process;}}}
   First_Come_First_Serve(name_of_process,burst_time,priority,number_of_processes);}
int main(){
   int number_of_processes;
   std::cout<<"Enter the number of processes: ";
   std::cin>>number_of_processes;
   std::string name_of_process[number_of_processes];
   int burst_time[number_of_processes];
   int priority[number_of_processes];
   for (int i=0;i<number_of_processes;i++){
      std::cout<<"Enter Name of Process:: ";
      std::cin>>name_of_process[i];
      std::cout<<"Enter Burst Time of Process "<<name_of_process[i]<<":: ";
      std::cin>>burst_time[i];
      std::cout<<"Enter Priority of Process "<<name_of_process[i]<<":: ";
      std::cin>>priority[i];}
   Sort(name_of_process,burst_time, priority,number_of_processes);}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Non Preemptive Priority'
Enter the number of processes: 4
Enter Name of Process:: p1
Enter Burst Time of Process p1:: 5
Enter Priority of Process p1:: 3
Enter Name of Process:: p2
Enter Burst Time of Process p2:: 6
Enter Priority of Process p2:: 2
Enter Name of Process:: p3
Enter Burst Time of Process p3:: 2
Enter Priority of Process p3:: 4
Enter Name of Process:: p4
Enter Burst Time of Process p4:: 7
Enter Priority of Process p4:: 1
Name Of Process Burst Time      Priority        Waiting Time    TurnAround Time
p4              7               1               0               7
p2              6               2               7               13
p1              5               3               13              18
p3              2               4               18              20

Average Waiting Time is:: 9.5
Average TurnAround Time is::14.5
```

10. Write a program to implement SRTF scheduling algorithm.

Code:-

```
#include <iostream>
void display(std::string name_of_process[], int burst_time[], int arrival_time[],int number_of_processes,int
waiting_time[],int turnaround_time[]){
   double total_waiting_time = 0;
   double total_turnaround_time = 0;
   std::cout<<"Name Of Process\tBurst Time\tArrival Time\tWaiting Time\tTurnAround Time"<<std::endl;
   for (int num=0;num<number_of_processes;num++){
      total_waiting_time += waiting_time[num];
      total_turnaround_time += turnaround_time[num];
std::cout<<name_of_process[num]<<"\t\t"<<burst_time[num]<<"\t\t"<<arrival_time[num]<<"\t\t"<<waiting_ti
me[num]<<"\t\t"<<turnaround_time[num]<<std::endl;}
   std::cout<<"\nAverage Waiting Time is:: "<<total_waiting_time / number_of_processes<<"\nAverage
TurnAround Time is:: "<<total_turnaround_time / number_of_processes<<std::endl;}
void shortest_remaining_time_first(std::string name_of_process[], int number_of_processes, int burst_time[],
int arrival_time[]){
   int waiting_time[number_of_processes];
   int turnaround_time[number_of_processes];
   int remaining_time[number_of_processes];
   std::copy(burst_time,burst_time+number_of_processes,remaining_time);
   int clock = 0;
   int completed = 0;
   float min_burst = float('inf');
   signed int shortest = -1;
   bool finished = false;
   while (completed != number_of_processes){
      for (int num=0;num<number_of_processes;num++){
         if (arrival_time[num] <= clock and remaining_time[num] < min_burst and remaining_time[num] > 0){
            min_burst = remaining_time[num];
            shortest = num;
            finished = true;}}
      if (not finished){
         clock += 1;
         continue;}
      remaining_time[shortest] -= 1;
      min_burst = remaining_time[shortest];
      if (min_burst == 0){
         min_burst = float('inf');}
      if (remaining_time[shortest] == 0){
         completed += 1;
         finished = false;
         int finish_time = clock + 1;
         waiting_time[shortest] = finish_time - burst_time[shortest] - arrival_time[shortest];
         turnaround_time[shortest] = finish_time - arrival_time[shortest];}
         if (waiting_time[shortest] < 0){
            waiting_time[shortest] = 0;}
      clock += 1;}
   display(name_of_process, burst_time, arrival_time, number_of_processes,waiting_time,turnaround_time);}
```

```
int main(){
   int number_of_processes;
   std::cout<<"Enter the number of processes: ";
   std::cin>>number_of_processes;
   std::string name_of_process[number_of_processes];
   int burst_time[number_of_processes];
   int arrival_time[number_of_processes];
   for (int i=0;i<number_of_processes;i++){
      std::cout<<"Enter Name of Process:: ";
      std::cin>>name_of_process[i];
      std::cout<<"Enter Burst Time of Process "<<name_of_process[i]<<":: ";
      std::cin>>burst_time[i];
      std::cout<<"Enter Arrival Time of Process "<<name_of_process[i]<<":: ";
      std::cin>>arrival_time[i];}
   shortest_remaining_time_first(name_of_process, number_of_processes, burst_time,arrival_time);}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Shortest Remaining Time Fir
st'
Enter the number of processes: 4
Enter Name of Process:: p1
Enter Burst Time of Process p1:: 4
Enter Arrival Time of Process p1:: 0
Enter Name of Process:: p2
Enter Burst Time of Process p2:: 2
Enter Arrival Time of Process p2:: 1
Enter Name of Process:: p3
Enter Burst Time of Process p3:: 3
Enter Arrival Time of Process p3:: 3
Enter Name of Process:: p4
Enter Burst Time of Process p4:: 9
Enter Arrival Time of Process p4:: 0
Name Of Process Burst Time      Arrival Time    Waiting Time    TurnAround Time
p1              4               0               2               6
p2              2               1               0               2
p3              3               3               3               6
p4              9               0               9               18

Average Waiting Time is:: 3.5
Average TurnAround Time is:: 8
```

11. Write a program to calculate sum of n numbers using Pthreads. A list of n numbers is divided into two smaller list of equal size, two separate threads are used to sum the sub lists.

Code:-

```
#include <pthread.h>
#include <iostream>
#include <vector>
using namespace std;
struct ThreadData {
   vector<int>& arr;
   int start;
   int end;
   int sum;};
void* calculate_sum(void* arg) {
   ThreadData* data = static_cast<ThreadData*>(arg);
   data->sum = 0;
   for (int i = data->start; i < data->end; ++i) {
      data->sum += data->arr[i];}
   pthread_exit(nullptr);}
int main() {
   int n;
   cout << "Enter the number of elements: ";
   cin >> n;
   vector<int> arr(n);
   cout << "Enter " << n << " elements: " << endl;
   for (int i = 0; i < n; ++i) {
      cin >> arr[i];}
   int mid = n / 2;
   ThreadData data1 = {arr, 0, mid, 0};
   ThreadData data2 = {arr, mid, n, 0};
   pthread_t thread1, thread2;
   pthread_create(&thread1, nullptr, calculate_sum, static_cast<void*>(&data1));
   pthread_create(&thread2, nullptr, calculate_sum, static_cast<void*>(&data2));
   pthread_join(thread1, nullptr);
   pthread_join(thread2, nullptr);
   int total_sum = data1.sum + data2.sum;
   cout << "The sum of the numbers is: " << total_sum << endl;
   return 0;}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Calculate sum of n numbers
using Pthreads. A list of n numbers is divided into two smaller  list of equal size, two separate threads are used to sum t
he sub lists'
Enter the number of elements: 6
Enter 6 elements:
1 2 3 4 6 8
The sum of the numbers is: 24
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$
```

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

Code:-

```
#include <iostream>
#include <vector>
#include <string>
#include <sstream>
using namespace std;
void display(const vector<pair<string, int>>& memory) {
   int count = 1;
   cout << "Memory Status:" << endl;
   for (const auto& frame : memory) {
      cout << "Frame: " << count << " | Process: " << frame.first << " | Size: " << frame.second << endl;
      count++;}}
void firstFit(int noOfFrames, vector<pair<string, int>>& memory, const pair<string, int>& process) {
   bool flag = false;
   for (int i = 0; i < noOfFrames; i++) {
      if (memory[i].first == "free" && memory[i].second >= process.second) {
         memory[i].first = process.first;
         display(memory);
         flag = true;
         break;}}
   if (!flag) {
      cout << "\nYou Have Not Enough Space To Run New Process" << endl;}}
void bestFit(int noOfFrames, vector<pair<string, int>>& memory, const pair<string, int>& process) {
   int flag = -1;
   for (int i = 0; i < noOfFrames; i++) {
      if (memory[i].first == "free" && memory[i].second >= process.second) {
         if (flag == -1 || memory[i].second < memory[flag].second) {
            flag = i;}}}
   if (flag != -1) {
      memory[flag].first = process.first;
      display(memory);
   } else {
      cout << "\nYou do not have enough space to run the new process." << endl;}}
void worstFit(int noOfFrames, vector<pair<string, int>>& memory, const pair<string, int>& process) {
   int flag = -1;
   for (int i = 0; i < noOfFrames; i++) {
      if (memory[i].first == "free" && memory[i].second >= process.second) {
         if (flag == -1 || memory[i].second > memory[flag].second) {
            flag = i;}}}
   if (flag != -1) {
      memory[flag].first = process.first;
      display(memory);
   } else {
      cout << "\nYou do not have enough space to run the new process." << endl;}}
int main() {
   vector<pair<string, int>> memory;
   while (true) {
      cout << "Enter memory type (used/free) and value, or 'q' to quit: ";
```

```
        string input;
        getline(cin, input);
        if (input == "q") {
            break;}
        istringstream iss(input);
        string memoryType;
        int value;
        iss >> memoryType >> value;
        memory.push_back({memoryType, value});}
    cout << "Enter process name and value: ";
    string processName, processValue;
    cin >> processName >> processValue;
    int processValueInt = stoi(processValue);
    pair<string, int> process = {processName, processValueInt};
    cout << "Main Menu \n1. First Fit\n2. Best Fit\n3. Worst Fit\nEnter your Choice:: ";
    int choice;
    cin >> choice;
    if (choice == 1) {
        firstFit(memory.size(), memory, process);
    } else if (choice == 2) {
        bestFit(memory.size(), memory, process);
    } else {
        worstFit(memory.size(), memory, process);}
    return 0;}
```

Output:-

```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Dynamic Storage Allocation'
Enter memory type (used/free) and value, or 'q' to quit: used 30
Enter memory type (used/free) and value, or 'q' to quit: free 40
Enter memory type (used/free) and value, or 'q' to quit: used 30
Enter memory type (used/free) and value, or 'q' to quit: free 50
Enter memory type (used/free) and value, or 'q' to quit: q
Enter process name and value: p1 30
Main Menu
1. First Fit
2. Best Fit
3. Worst Fit
Enter your Choice:: 3
Memory Status:
Frame: 1 | Process: used | Size: 30
Frame: 2 | Process: free | Size: 40
Frame: 3 | Process: used | Size: 30
Frame: 4 | Process: p1 | Size: 50
```


```
ratneshkumar@RATNESH-KUMAR:/mnt/c/Users/user/Documents/GitHub/Operating System/Practical/C++$ ./'Dynamic Storage Allocation'
Enter memory type (used/free) and value, or 'q' to quit: used 30
Enter memory type (used/free) and value, or 'q' to quit: free 40
Enter memory type (used/free) and value, or 'q' to quit: used 30
Enter memory type (used/free) and value, or 'q' to quit: free 50
Enter memory type (used/free) and value, or 'q' to quit: q
Enter process name and value: p1 30
Main Menu
1. First Fit
2. Best Fit
3. Worst Fit
Enter your Choice:: 3
Memory Status:
Frame: 1 | Process: used | Size: 30
Frame: 2 | Process: free | Size: 40
Frame: 3 | Process: used | Size: 30
Frame: 4 | Process: p1 | Size: 50
```